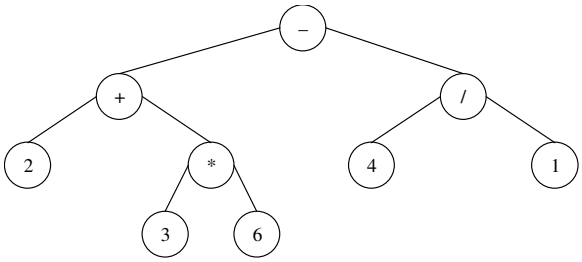
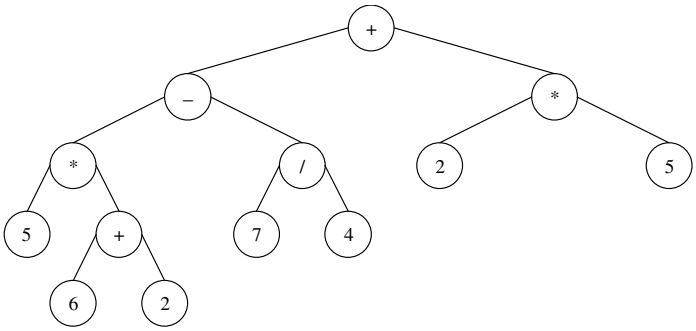
ARBEITSBLATT ZU TERMBÄUMEN

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

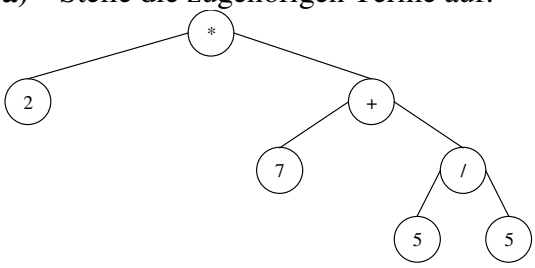


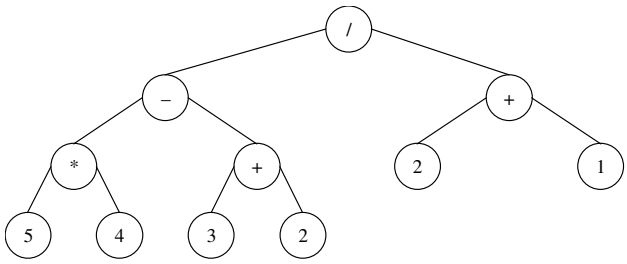
Beispiel 2: 5 * (6 + 2) – 7 / 4 + 2 * 5



Im folgenden wist du ein Delphi-Projekt erstellen, welches Termbäume aufbauen, umformen und ausrechnen kann. Doch zuerst ein paar kleine Übungen zum Aufwärmen.

a) Stelle die zugehörigen Terme auf:





b) Stelle die zugehörigen Termbäume zu den Termen **(I)** 7 * 6 – (3 * 5 – 2 + 1) und **(II)** 3 – 8 / 4 * 3 + 5 auf.

c) Durchläuft man einen Termbaum in Preorder, so erhält man den sogenannten *Präfix-Term*, durchläuft man ihn in Postorder, so erhält man den *Postfix-Term*.
Gib sowohl den Präfix-Term als auch den Postfix-Term der Beispiel-Termbäume an (***Beispiel 1*** und ***2***).

d) Gegeben ist nun der Präfix-Term **(I)** – * 2 + 2 3 + 6 1. Stelle den zugehörigen Termbaum auf. Versuche das gleiche mit den Präfix-Termen **(II)** * + 8 7 – 7 / 4 2 und **(III)** + + + 1 1 1 + + 1 1 + 1 1.

e) Gegeben ist nun der Postfix-Term **(I)** 1 3 5 * + 6 – 8 3 * –. Stelle den zugehörigen Termbaum auf. Versuche das gleiche mit den Postfix-Termen **(II)** 8 7 7 * + 4 2 / – und **(III)** 1 1 + 1 + 1 1 + 1 + +.

Und jetzt, wo du dich mit der Struktur auskennst, sollst du ein Programm schreiben, welches mit dieser Datenstruktur sinnvoll umgehen kann. D. h. im einzelnen, dass das Programm folgende Aufgaben erledigen kann (nach Schwierigkeitsgrad geordnet):

- Es kann einen vorhanden Termbaum grafisch anzeigen (Unit BaumOut.Pas)
- Es kann einen vorhanden Termbaum als Präfix-, Postfix- oder Infix-Term ausgeben.
- Es kann einen Termbaum arithmetisch auswerten.
- Es kann einen Präfixterm in einen Termbaum überführen (noch relativ leicht).
- Es kann einen Postfixterm in einen Termbaum überführen (etwas komplizierter).
- Es kann einen Infixterm in einen Termbaum überführen (schwer aber machbar).

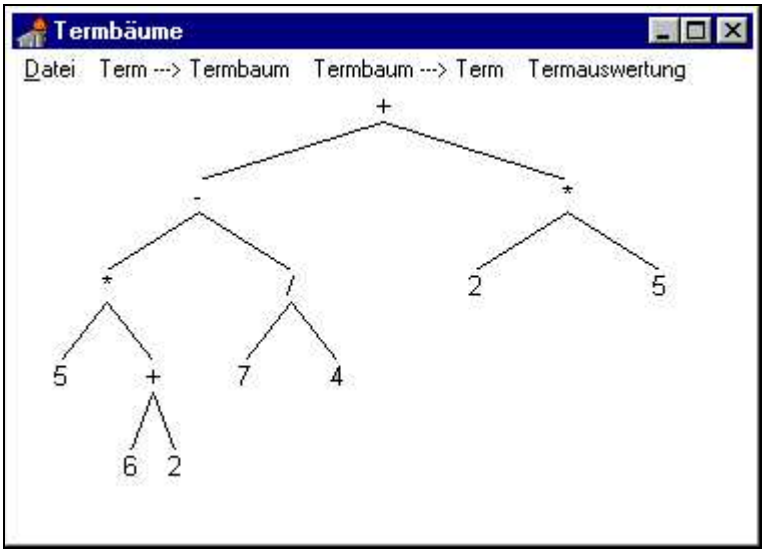
Eine Datenstruktur für diese Problem habe ich rechts abgedruckt. Allerdings ist diese nicht zwingend erforderlich.

```
unit TermBaum;

interface

uses BinBaum

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


type TTermbaum = class(TBinBaum)
       Inhalt: char;                                    { nur Ziffern als Operanden erlaubt              }
       constructor create(z: char);                     { Konstruktor mit Übergabe des Wurzelinhalts     }
       function WurzelinhaltToString: string; override; { Aus TBinbaum geerbt                            }
       function ToPreOrder: string;                     { Gibt den Termbaum als PreOrder-String aus      }
       function ToInOrder: string;                      { Gibt den Termbaum als InOrder-String aus       }
       function ToPostOrder: string;                    { Gibt den Termbaum als PostOrder-String aus     }
       function Auswertung: Integer;                    { Wertet den Termbaum arithmetisch aus           }
     end;

function FromPreOrder(var Zeichenkette: string): TTermBaum;  { Erzeugung eines Termbaums aus PreOrder-String   }
function FromInOrder(var Zeichenkette: string): TTermbaum;   { Erzeugung eines Termbaums aus InOrder-String    }
function FromPostOrder(var Zeichenkette: string): TTermbaum; { Erzeugung eines Termbaums aus PostOrder-String  }
```

Dein Projekt wird vermutlich die folgenden Units benötigen:

Grunddatentyp Binärbaum:	Unit BinBaum.Pas
Baumausgabe:	Unit BaumOut.Pas
Termbaum-Definition:	Unit TermBaum.Pas
Oberflächen-Definition:	Unit U_IO.Pas

Eine mögliche Oberfläche ist die folgende:



Viel Spaß bei der Arbeit!