

Die Datenstruktur Binärbaum

Beispiele für Baumstrukturen

Hierarchische Strukturen, die beginnend von einem Startpunkt beliebig weit verzweigen, bezeichnet man in der Informatik als Baum. Kennzeichnend für einen Baum ist, dass es keine zyklischen Verbindungen zwischen den Elementen gibt. Ein Beispiel für eine solche hierarchische Struktur ist die Ordnerstruktur eines Dateisystems. Abbildung 1 zeigt die Modellierung einer exemplarischen Ordnerstruktur als Baum.

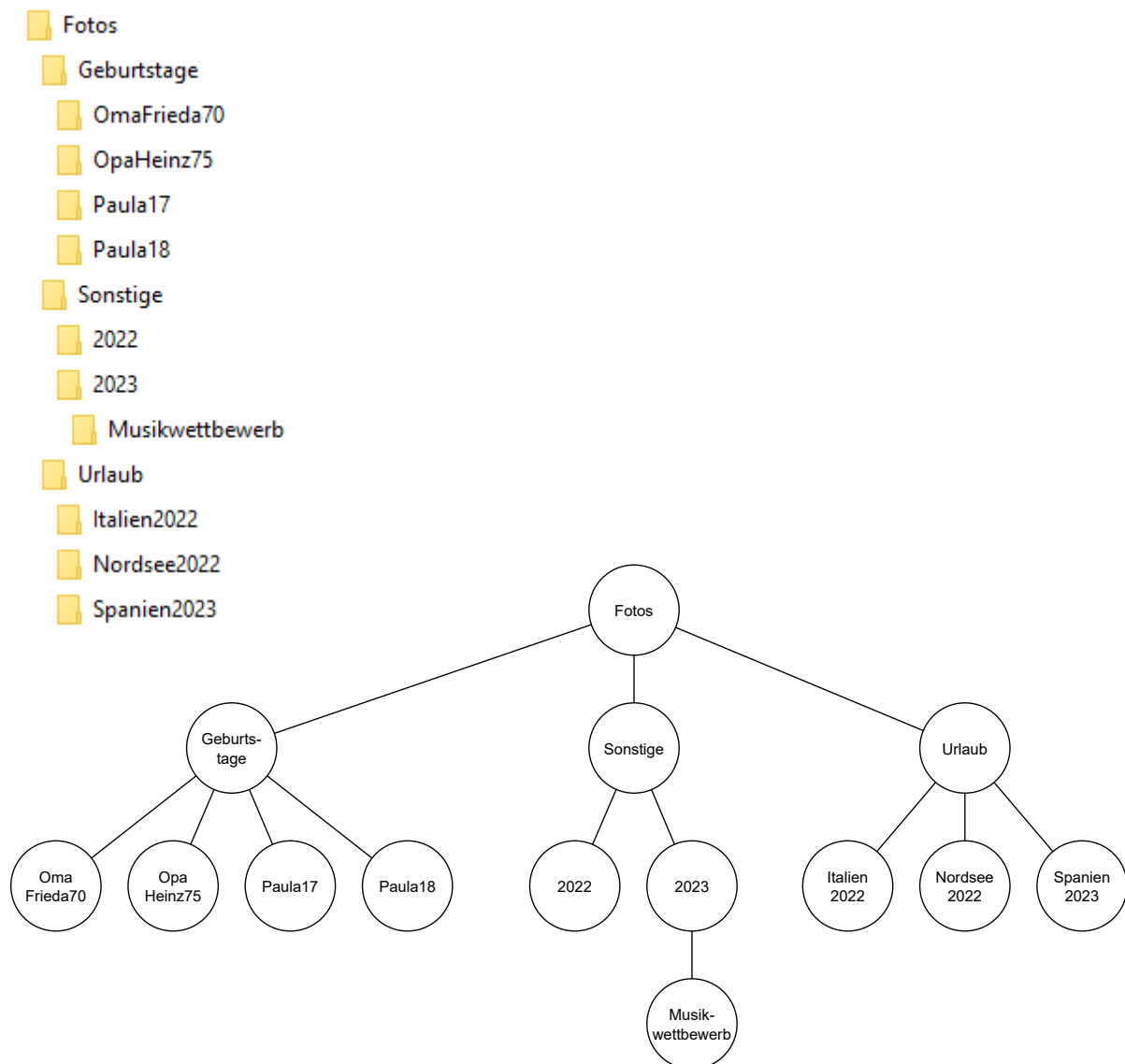


Abbildung 1: Modellierung einer exemplarischen Ordnerstruktur als Baum

Aufgabe 1: Sammeln Sie weitere Beispiele für hierarchische Strukturen, die sich als Baum darstellen lassen. Zeichnen Sie einen entsprechenden Baum.

Bäume in der Informatik

Die einzelnen Elemente eines Baums bezeichnet man als Knoten. Für spezielle Knoten wurden weitere Fachbegriffe eingeführt, teilweise in Analogie zu Bäumen in der Natur: So bezeichnet man den Startknoten, von dem aus alle weiteren Knoten abzweigen, als **Wurzel**. Anders als bei Bäumen in der Natur zeichnet man die Wurzel in der Informatik nach oben und der Baum wächst von dort aus nach unten. Die nachfolgenden, abzweigenden Knoten bezeichnet man als **Kindknoten**, den Vorgänger als **Vater**- oder Elternknoten. Die Wurzel ist der einzige Knoten, der keinen Vorgänger und damit keinen Vaterknoten hat. Einen Knoten, der keine Kinder hat, bezeichnet man als **Blatt**. Alle Knoten, die mindestens ein Kind haben sind die **inneren Knoten**. Graphisch lassen sich die Knoten in mehreren Ebenen anordnen. Die Anzahl an Ebenen bezeichnet man als **Höhe** des Baumes. Sie entspricht also der Anzahl der Knoten des längsten Pfades von der Wurzel zu einem Blatt¹. Die Verbindung zwischen zwei Knoten ist eine **Kante**.

Aufgabe 2:

- Beschriften Sie den Baum in Abbildung 2 mit den passenden Fachbegriffen.
- Malen Sie die Knoten in der jeweils passenden Farbe an.
rot: Wurzel, blau: innerer Knoten, grün: Blatt
- Tragen Sie die Werte „a“, „b“, „c“, „d“, und „e“ so in Knoten des Baums ein, dass der Knoten mit dem Wert „d“, einen Elternknoten mit dem Wert „e“ und Kindknoten mit den Werten „a“, „b“ und „c“ hat.

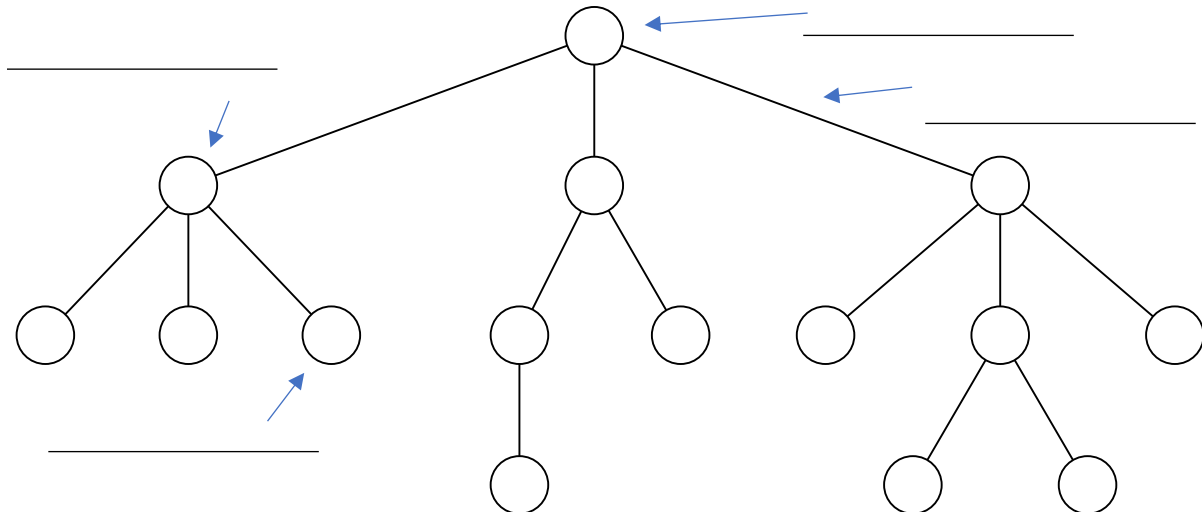


Abbildung 2: Struktur eines Baums für Aufgabe 2

¹ In der Literatur wird als Höhe eines Baumes entweder die Anzahl der Knoten oder die Anzahl der Kanten des längsten Pfades von der Wurzel bis zu einem Blatt bezeichnet. Die Höhe des Baumes in Abbildung 2 wäre somit 4 oder 3.

Aufgabe 3:

- a) Domainnamen von Webseiten sind hierarchisch organisiert. Geben Sie alle Domainnamen an, die in dem Binärbaum in Abbildung 3 dargestellt sind.

Beispiel: informatik.tu.uni.org

- b) Geben Sie für den Baum in Abbildung 3 die Wurzel, die Blattknoten und die Höhe an.

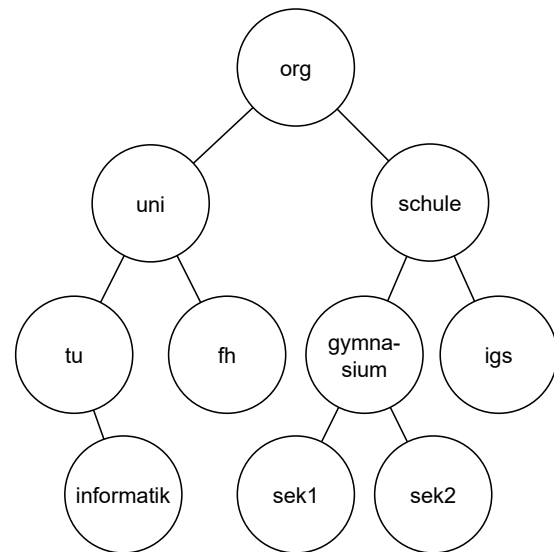


Abbildung 3: Baumdarstellung von Domainnamen

Binärbäume

Im Folgenden beschäftigen wir uns mit Bäumen, bei denen jeder Knoten maximal zwei Kinder haben kann. Solche Bäume bezeichnet man als Binärbäume. Die Kinder unterscheidet man in linkes Kind und rechts Kind.

Aufgabe 4:

- a) Entscheiden Sie für die Bäume in den Abbildungen 1 bis 3 jeweils, ob es sich um einen Binärbaum handelt.
- b) Betrachten Sie noch einmal die Beispiele, die Sie in Aufgabe 1 gesammelt haben. In welchen Fällen eignet sich ein Binärbaum für die Modellierung?

Traversierung von Binärbäumen

Da es sich bei einem Binärbaum nicht um eine lineare Struktur handelt, gibt es verschiedene Möglichkeiten, in welcher Reihenfolge die Knoten nacheinander ausgelesen werden.

Abbildung 4 zeigt die Darstellung des Rechenterms $(3 + 7) * 2 - (4 * 6)$ als Binärbaum.

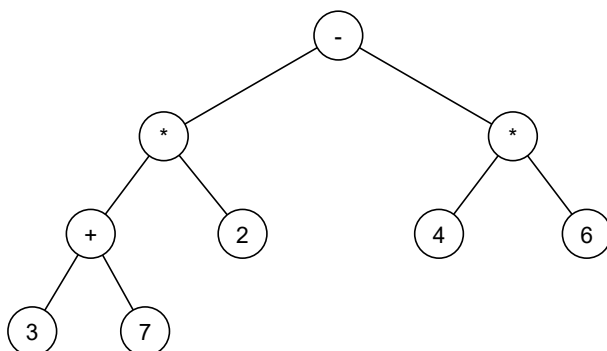


Abbildung 4: Termbaum

Eine Möglichkeit den Binärbaum zu durchlaufen, um den Term zu rekonstruieren, wäre die sogenannte **inorder**-Reihenfolge. Dabei wird zunächst der linke Teilbaum, dann die Wurzel und schließlich der rechte Teilbaum ausgelesen. Dieses Prinzip wird rekursiv auf den linken und rechten Teilbaum angewendet, bis schließlich die Blätter ausgelesen wurden. Als Term ergibt sich somit:

$3 + 7 * 2 - 4 * 6$. Um die Information, in welcher Reihenfolge die Rechenoperationen ausgeführt werden sollen, nicht zu verlieren, müssten hier konsequent Klammern um die Teilterme, die sich aus den Teilbäumen ergeben, gesetzt werden: $((3 + 7) * 2) - (4 * 6)$

Bei der Auswertung von Rechentermen kommen daher manchmal auch andere Schreibweisen bzw. Reihenfolgen zum Einsatz: Durchläuft man den Binärbaum in **preorder**-Reihenfolge, wird rekursiv zuerst die Wurzel, dann der linke Teilbaum und dann der rechte Teilbaum besucht. Damit ergibt sich die folgende Ausgabe des Terms, die auch als Präfix-Schreibweise bezeichnet wird: $- * + 3 7 2 * 4 6$

Alternativ kann der Binärbaum in **postorder**-Reihenfolge durchlaufen werden, indem rekursiv zuerst der linke Teilbaum, dann der rechte Teilbaum und schließlich die Wurzel besucht werden. Damit ergibt sich die folgende Ausgabe des Terms, die auch als Postfix-Schreibweise bezeichnet wird:

$3 7 + 2 * 4 6 * -$

Sowohl bei der Präfix- als auch bei der Postfix-Schreibweise ist die Reihenfolge der Auswertung ohne Klammersetzung eindeutig.

Das systematische Durchlaufen eines Baumes bezeichnet man auch als **Traversierung**. Die verschiedenen Möglichkeiten der Traversierung: inorder, preorder bzw. postorder lassen sich auf beliebige Binärbäume anwenden. Ob eine bestimmte Reihenfolge sinnvoller ist als andere, hängt vom Kontext ab. In jedem Fall wird bei der systematischen Traversierung kein Knoten vergessen.

Als vierte Möglichkeit können die Knoten in **levelorder**-Reihenfolge durchlaufen werden. Dabei werden die Knoten beginnend bei der Wurzel Ebene für Ebene ausgegeben. Für den Termbaum würde sich die Reihenfolge $- * * + 2 4 6 3 7$ ergeben. Im Kontext der Auswertung eines Rechenterms ist diese Reihenfolge aber nicht besonders hilfreich.

Aufgabe 5: Geben Sie den Stammbaum in Abbildung 5 in *inorder*, *preorder*, *postorder* und *levelorder* Reihenfolge aus.

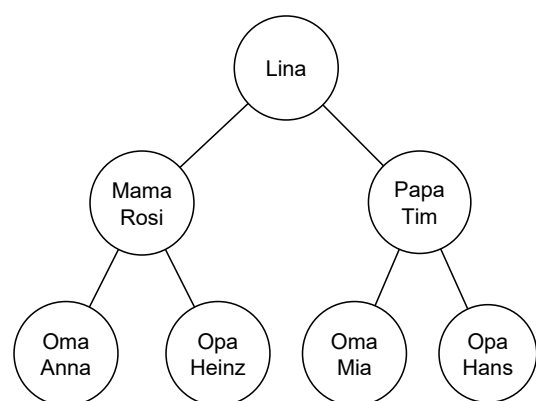


Abbildung 5: Stammbaum von Lina

Umsetzung eines Binärbaums als rekursive Datenstruktur

Aufgrund der nicht linearen Struktur eignet sich für die Umsetzung eines Binärbaums als Datenstruktur ein rekursiver Ansatz. Demnach besteht ein Binärbaum aus einer Wurzel sowie einem linken Teilbaum und einem rechten Teilbaum. Der linke und / oder der rechte Teilbaum können ein leerer Baum sein. Andernfalls lassen sich der linke und der rechte Teilbaum wieder als Binärbaum beschreiben, der aus einer Wurzel, einem linken und einem rechten Teilbaum besteht. Abbildung 6 veranschaulicht diese Betrachtungsweise. Aus der Sicht des gelben Knotens, der Wurzel, ist der linke Teilbaum jeweils blau und der rechte Teilbaum jeweils grün gefärbt. Diese Betrachtung kann von Ebene zu Ebene in Richtung Blattknoten rekursiv fortgesetzt werden, bis ein Blatt schließlich ein Wurzelknoten mit einem leeren linken und rechten Teilbaum ist.

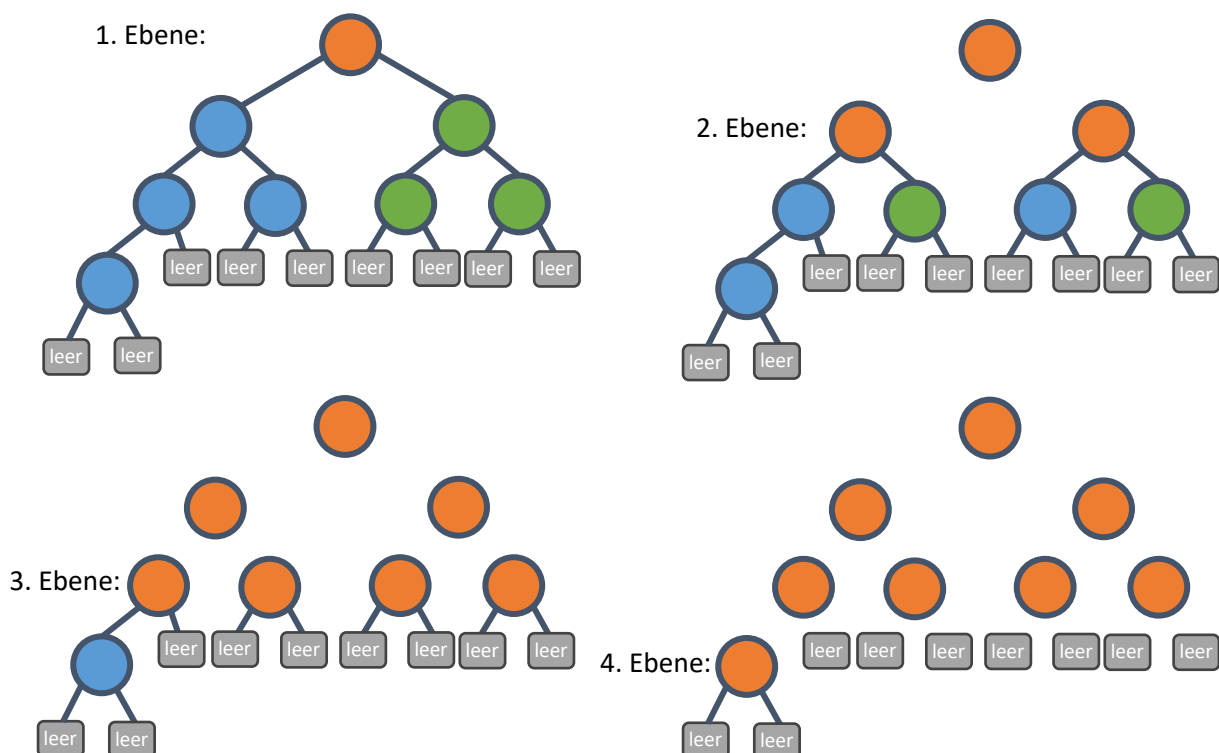


Abbildung 6: Binärbaum als rekursive Datenstruktur

Im Anhang finden Sie eine Übersicht über die Operationen einer Klasse `Binärbaum`, die eine rekursive Implementierung der Datenstruktur Binärbaum zur Verfügung stellt. Der Standardkonstruktor `BinTree()` erzeugt einen leeren Binärbaum. Ein leerer Binärbaum besitzt keinen Inhalt und keine Teilbäume. Der Konstruktor `BinTree(inhalt: Inhaltstyp)` erzeugt einen Binärbaum mit dem Wert des Parameters `inhalt` als Inhalt des Wurzelknotens². Außerdem erhält die Wurzel jeweils einen leeren Binärbaum als linkes und rechtes Kind.

Mit den Operationen `getItem` bzw. `setItem` kann der Inhaltswert der Wurzel ausgelesen bzw. neu gesetzt werden. Die Operation `getItem` darf nur angewendet werden, wenn die Wurzel einen Inhaltswert hat, das heißt, wenn der Baum nicht leer ist. Daher kann mit der Operation `isEmpty` ggf. vorher geprüft werden, ob ein Baum leer ist. Bei einem leeren Baum setzt `setItem` den Inhalt

² Der Inhaltstyp kann je nach Kontext festgelegt werden, so dass die Knoten eines Binärbaums z. B. Zeichenketten, Zahlen oder auf Objektvariablen aufnehmen können.

der Wurzel auf den übergebenen Wert und erzeugt einen leeren linken und rechten Teilbaum. Ist der Baum nicht leer, wird der Inhalt der Wurzel überschrieben.

Auf den linken bzw. rechten Teilbaum eines Knotens kann mithilfe der Operationen `getLeft` bzw. `getRight` zugegriffen werden. Diese geben den gesamten linken bzw. rechten Teilbaum zurück. Das kann ggf. auch ein leerer Baum sein. Auch diese Operationen dürfen aber nicht auf einen leeren Baum angewendet werden, da hier weder ein linker noch rechter Teilbaum existieren. Es muss also ggf. vorher mit der Operation `isEmpty` geprüft werden, ob der Baum leer ist. Zum Setzen bzw. Überschreiben des linken bzw. rechten Teilbaums stehen die Methoden `setLeft` und `setRight` zur Verfügung.

Mithilfe der Operation `isLeaf` kann überprüft werden, ob es sich bei einem Knoten um ein Blatt handelt. Mit der Operation `setEmpty` kann ein Binärbaum schließlich auf einen leeren Binärbaum zurückgesetzt werden.

Merke: Dadurch, dass ein nicht leerer Binärbaum nicht nur einen Inhalt, sondern auch einen linken und einen rechten Teilbaum hat, reicht es einmal mit der Operation `isEmpty` zu prüfen, ob der Baum leer ist, um andernfalls für diesen die Operationen `getItem`, `getLeft` und `getRight` ausführen zu können, ohne dass es zu einer Fehlermeldung kommt. Die Operationen `getLeft` und `getRight` geben in diesem Fall einen Binärbaum zurück, dieser kann jedoch leer sein. Nur ein leerer Baum hat weder einen Inhalt noch einen linken und rechten Teilbaum.

Aufgabe 6:

- a) Erstellen Sie ein Programm `StammbaumApp`, welches den Stammbaum in Abbildung 5 in einer globalen Variablen `stammbaum` vom Typ `BinTree` verwaltet. Verwenden Sie die Operationen der Klasse `BinTree`, um den Stammbaum beim Starten des Programms zu erzeugen. Sie können als Inhaltstyp der Knoten den Datentyp `Zeichenkette` verwenden.

Hinweis: Der Ordner *Vorlage_Aufgabe6* enthält eine Vorlage für das Programm. Hier wurde bereits ein Binärbaum mit Knoten für „Lina“ und „Mama Rosi“ angelegt.

- b) Abbildung 7 zeigt die rekursive Implementierung einer Operation `traversiere`, die den Inhalt aller Knoten des übergebenen Baums ausgibt. Untersuchen Sie, in welcher Traversierungsreihenfolge die Ausgabe erfolgt.

```
1 public String traversiere(BinTree b) {
2     String ausgabe = "";
3     if(!b.isEmpty()) {
4         ausgabe = ausgabe + traversiere(b.getLeft());
5         ausgabe = ausgabe + b.getItem() + "\n";
6         ausgabe = ausgabe + traversiere(b.getRight());
7     }
8     return ausgabe;
9 }
```

Abbildung 7: Implementierung einer Methode `traversiere`

- c) Ergänzen Sie in Ihrem Programm `StammbaumApp` Operationen `inorder`, `preorder`, `postorder` und `levelorder`, die jeweils einen Binärbaum als Parameter erhalten und eine Zeichenkette mit dem Inhalt aller Knoten in der entsprechenden Reihenfolge zurückgeben. Verwenden Sie die Operationen, um Anwendenden die Möglichkeit zu geben, den Stammbaum in den verschiedenen Traversierungsreihenfolgen ausgeben zu lassen.

Tipps:

- Ergänzen Sie nach jedem Inhaltswert einen Zeilenumbruch ("`\n`"), um die Inhalte übersichtlich anzeigen zu können.
- Für die Traversierung in levelorder ist es hilfreich, die Datenstruktur Schlange vom Inhaltstyp Binärbaum zu verwenden.

Aufgabe 7:

- Legen Sie eine Klasse `Person` an, welche mindestens die Attribute `Name`, `Titel`, `Geburtsjahr` enthält. Ergänzen Sie geeignete Operationen, z. B. für die Ausgabe der Attributwerte in einer Zeichenkette.
 - Ändern Sie Ihr Programm aus Aufgabe 6 so, dass die Knoten des Stammbaums Objekte vom Typ `Person` enthalten.
 - Erweitern Sie Ihr Programm so, dass es die Möglichkeit gibt, nach einer Person zu suchen und die Eltern ausgeben zu lassen.
 - Erweitern Sie Ihr Programm so, dass weitere Vorfahren in dem Stammbaum ergänzt werden können. Erfragen Sie dazu vom Anwendenden die Person, für die ein Elternteil ergänzt werden soll und die Daten des Elternteils. Mütter werden als linkes Kind, Väter als rechtes Kind ergänzt. Wenn bereits ein entsprechender Knoten existiert, wird nur der Inhaltswert des Knotens geändert.
 - Ergänzen Sie in Ihrem Programm die Möglichkeit, ausgeben zu lassen, wie viele Generationen in dem Stammbaum enthalten sind. Die Generationen müssen nicht vollständig abgebildet sein.
- Tip:** Die Anzahl der Generationen entspricht der Höhe des Baumes.

Aufgabe 8: Der Verlauf eines Tennisturniers kann in einem Binärbaum abgebildet werden. Dabei enthalten die Blätter alle Spieler³. Zwei Spieler mit dem gleichen Vaterknoten tragen ein Match aus, wobei der Gewinner in den Vaterknoten eingetragen wird. So steht am Ende der Sieger des Turniers in der Wurzel des Baumes. Gehen Sie für die Aufgabenteile a) bis c) davon aus, dass das Turnier bereits beendet ist und alle Spieler bzw. Sieger in einem global definierten Binärbaum `turnierbaum` vom Inhaltstyp `Zeichenkette` vorliegen.

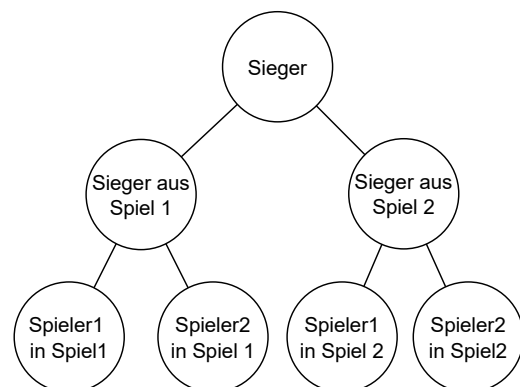


Abbildung 8: Darstellung eines Tennisturniers als Binärbaum

- Eine Operation `bestimmeAnzahlSpiele` soll für einen Spieler, der als Parameter vom Typ `Zeichenkette` übergeben wird, bestimme, wie viele Spiele er in dem Turnier gespielt hat, und die Anzahl als Ganzzahl zurückgeben.
- Eine Operation `zeigePaarungen(spielrunde: Ganzzahl): Zeichenkette` soll alle Paarungen einer Spielrunde ermitteln und in einer Zeichenkette zurückgeben. Jede Ebene des Baumes stellt eine Spielrunde dar. Erstellen Sie für die Operation `zeigePaarungen` ein Struktogramm.

³ Hier wird exemplarisch von einem Tennisturnier der Männer ausgegangen und daher nur die männliche Form verwendet. Analog kann das Turnier der Frauen modelliert werden.

Tipp: Nummeriert man die Knoten eines vollständigen Binärbaumes in levelorder-Reihenfolge beginnend mit 1 für die Wurzel, so haben die Knoten einer Ebene n die Nummern 2^n bis $2^{n+1}-1$.

- c) Eine Operation `sucheGegner` soll für einen Spieler, der als Parameter vom Typ `Zeichenkette` übergeben wird, alle Gegner zurückgeben, gegen die er während des Turniers gespielt hat. Die Rückgabe der Gegner erfolgt als `dynamische Reihung` vom Inhaltstyp `Zeichenkette`. Erstellen Sie für die Operation `sucheGegner` ein Struktogramm.
- d) Zu Beginn des Turniers sind nur die Blätter des Baumes mit den entsprechenden Spielern gefüllt. Alle inneren Knoten enthalten eine leere Zeichenkette. Eine Operation `trageSiegerEin` erhält als Parameter `Spieler1` und `Spieler2` sowie den Sieger als `Zeichenkette`. Die Operation soll den Sieger als Inhaltswert des Vaterknotens von `Spieler1` und `Spieler2` eintragen. Gehen Sie vereinfachend davon aus, dass `Spieler1` und `Spieler2` in der passenden Reihenfolge übergeben werden. Erstellen Sie für die Operation `trageSiegerEin` ein Struktogramm.

Operationen der Klasse Binärbaum⁴

`BinTree()`

Ein leerer Baum wird erzeugt. Er besitzt keinen Inhalt und keine Teilbäume.

`BinTree(inhalt: Inhaltstyp)`

Ein Baum wird erzeugt. Die Wurzel erhält den übergebenen Inhalt als Wert. Der Baum besitzt jeweils einen leeren Baum als linken und rechten Teilbaum.

`isEmpty(): Wahrheitswert`

Wenn der Baum ein leerer Baum ist, wird der Wert `wahr` zurückgegeben, sonst der Wert `falsch`.

`getItem(): Inhaltstyp`

Die Operation gibt den Inhaltswert der Wurzel des Baumes zurück.

`setItem(inhalt: Inhaltstyp)`

Die Wurzel des Baums erhält den übergebenen Inhalt als Wert. Bei einem leeren Baum wird zusätzlich als linker und rechter Teilbaum jeweils ein leerer Baum gesetzt.

`isLeaf(): Wahrheitswert`

Wenn der Baum jeweils einen leeren Baum als linken und rechten Teilbaum besitzt, also ein Blatt ist, wird der Wert `wahr` zurückgegeben, sonst der Wert `falsch`.

`getLeft(): Binärbaum`

Die Operation gibt den linken Teilbaum zurück.

`setLeft(b: Binärbaum)`

Der übergebene Baum wird als linker Teilbaum gesetzt.

`getRight(): Binärbaum`

Die Operation gibt den rechten Teilbaum zurück.

`setRight(b: Binärbaum)`

Der übergebene Baum wird als rechter Teilbaum gesetzt.

`setEmpty()`

Der Baum wird zu einem leeren Baum, d. h. er besitzt keinen Inhalt und keine Teilbäume.

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#). Von der Lizenz ausgenommen ist das InfSII-Logo.

Für die korrekte Ausführbarkeit der Quelltexte in diesem Arbeitsblatt wird keine Garantie übernommen. Auch für Folgeschäden, die sich aus der Anwendung der Quelltexte oder durch eventuelle fehlerhafte Angaben ergeben, wird keine Haftung oder juristische Verantwortung übernommen.

⁴ Die hier beschriebene Datenstruktur Binärbaum orientiert sich an den Vorgaben des Kerncurriculums für das Gymnasium – gymnasiale Oberstufe, die Gesamtschule – gymnasiale Oberstufe, das Kolleg für das Fach Informatik, Niedersächsisches Kultusministerium, 2017 sowie den Ergänzenden Hinweisen zum Kerncurriculum Informatik für die gymnasiale Oberstufe am Gymnasium und an der Gesamtschule sowie für das Kolleg, Niedersächsisches Kultusministerium, Neufassung Juni 2025.