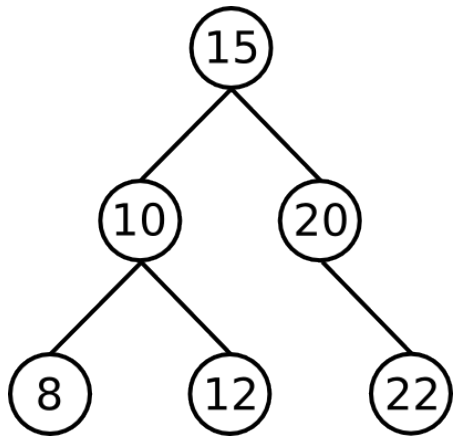
Ziel

- **Ziel:** Implementierung eines einfachen binären Suchbaums
- Ergänzung zu unseren Kurzeinstieg und Literaturreferenzen:
  - Vorlesung *“Algorithmen und Datenstrukturen”*
  - Wikipedia: Binärbaum
  - Wikibooks: Binärbäume in C

Binäre Suchbäume

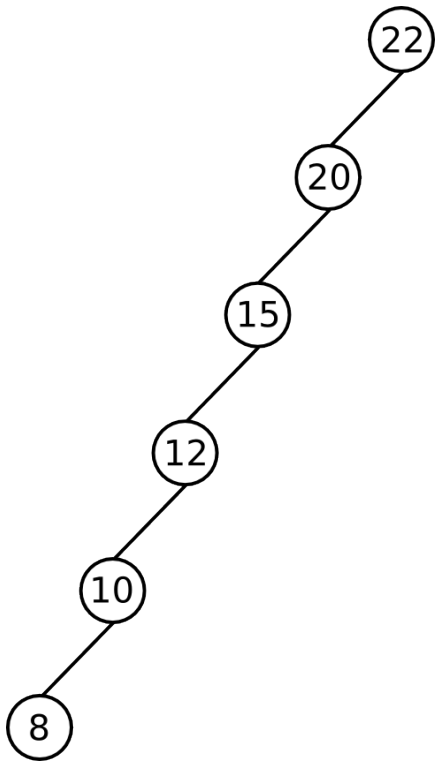
- Binärbaum ist gewurzelter Baum
- Jeder Knoten hat höchstens zwei Kindknoten
- Jeder linke und rechte Teilbaum ist wieder ein Binärbaum
- Für jeden Teilbaum mit Wurzel w gilt:
  - Alle Werte im linken Teilbaum sind kleiner als w
  - Alle Werte im rechten Teilbaum sind größer als oder gleich w



- Die Suchzeit von Elementen ist allerdings stark von der Anordnung im Baum abhängig!

Binäre Suchbäume - Worst Case

- Fügen wir die gleichen Knoten aus dem vorherigen Beispiel in einer anderen Reihenfolge ein. . .



. . . erhalten wir einen entarteten Baum.

Binäre Suchbäume - Varianten

- Um bessere Suchzeiten zu erreichen gibt es verschiedene Umsetzungen:
  - *Balancierter Baum:*

    Da die Laufzeiten der meisten Operationen auf Bäumen von deren Höhe abhängen, garantieren balancierte Bäume eine maximale Höhe von $c \cdot log(n)$. Diese Bäume können beispielsweise auch nach der Wahrscheinlichkeit ihrer Knotenzugriffe gewichtet und ausbalanciert worden sein.

  $\Rightarrow$ *AVL-Baum:*

    Der AVL-Baum (benannt nach Georgi Maximowitsch *A*delson-*V*elski und Jewgeni Michailowitsch *L*andis) ist ein höhenbalancierter Binärbaum bei dem sich die Höhe zweier benachbarter Teilbäume um höchstens eins unterscheidet. Dies wird erreicht indem der Baum beim Einfügen von Elementen neu balanciert wird (Stichwort: *Links- und Rechtsrotation*) und erlaubt alle Operationen in höchstens $O(log(n))$.

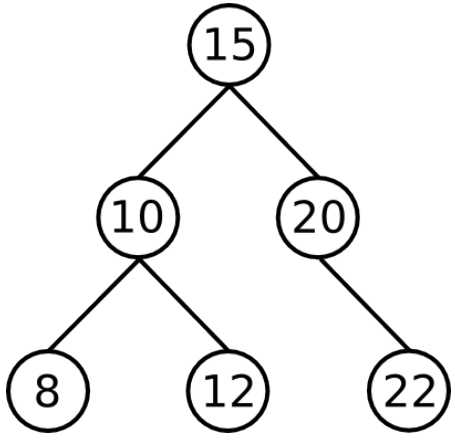
  - Es gibt noch viele Weitere . . .

- Es gibt verschiedene Möglichkeiten einen Binärbaum zu *traversieren*.
- *in-order* (= Reihenfolge der Werte im Baum):
  - linker Teilbaum → Wurzel → rechter Teilbaum

  ⇒ `8 10 12 15 20 22`
- *pre-order* (*Tiefensuche*):
  - Wurzel → linker Teilbaum → rechter Teilbaum

  ⇒ `15 10 8 12 20 22`
- *post-order*:
  - linker Teilbaum → rechter Teilbaum → Wurzel

  ⇒ `8 12 10 22 20 15`



Binäre Suchbäume - An den Rechner... fertig... los...

- **Jetzt:** Implementierung eines einfachen Binärbaums.
  ⇒ Implementierung der vordefinierten Header-Datei
  ⇒ Queue als Hilfstruktur gegeben

- Vergessen Sie nicht die grundlegenden Techniken zur Fehlervermeidung!

- Dokumentieren Sie ihren Code und erzeugen Sie mit *Doxygen* eine Dokumentation.
  ⇒ `doxywizard` *oder* `doxygen DOXYFILE`

- Suchen Sie mit Hilfe von *valgrind* nach Speicherlöchern
  ⇒ `valgrind -tool=memcheck -leak-check=yes ./tree`