

Datenverwaltung

Daten verwalten heisst

- **speichern**
 - ⇒ schnell und platzsparend
- **wiederfinden**
 - ⇒ schnell und zuverlässig
- entfernen, sortieren, kopieren, ...

Beispiele: Bücher, Explorer, Übung

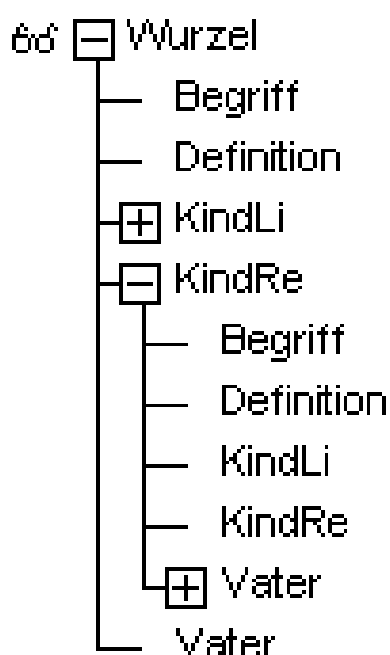
3 Bäume 57

3.1 Datenverwaltung 58

3.2 Baumstrukturen 59

3.2.1 Definitionen 59

3.2.2 Beispiel Wörterbuch.. 60



Stichwortverzeichnis

A

Ablaufstruktur 80,81,83

absolute Adresse 24

Baum und Binärbaum

Ein **Binärbaum** ist ein *geordneter* Baum, dessen Knoten *zwei* Nachfolger haben, einen *linken* und einen *rechten*.

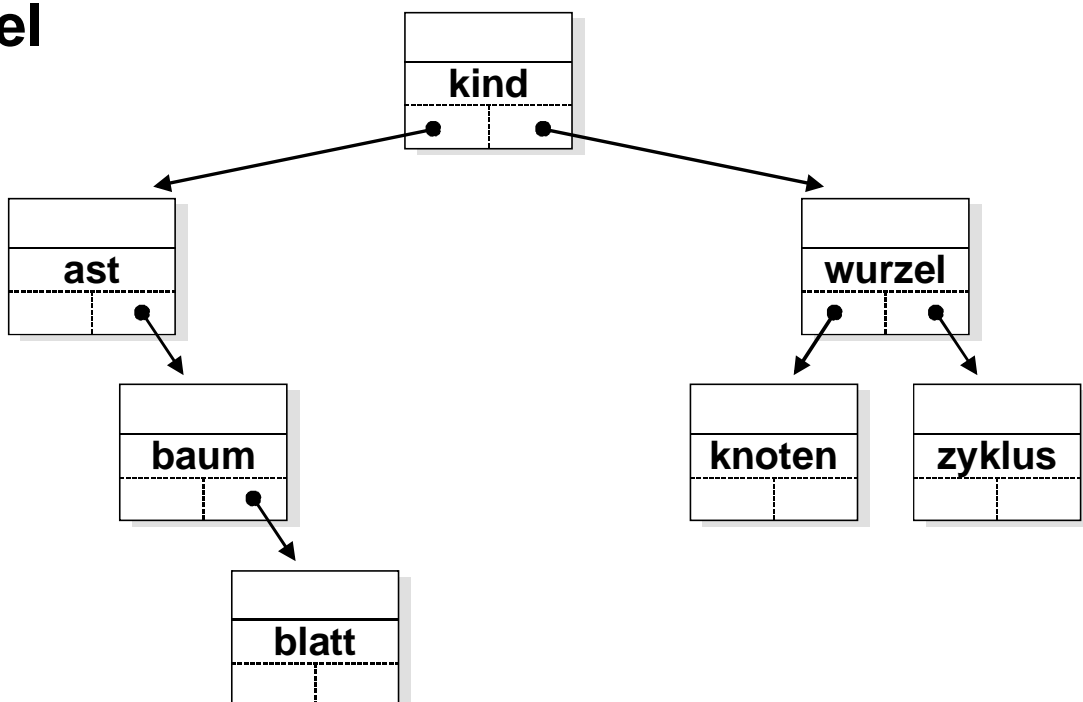
Ein Binärbaum heisst **sortiert**, wenn *für jeden* Knoten *k* gilt:
1. im *linken* Unterbaum sind alle Inhalte kleiner als in *k* und
2. im *rechten* Unterbaum sind alle Inhalte grösser als in *k*.

Die Anzahl Verweise, denen man maximal folgen muss, um zu einem Blatt zu gelangen, heisst **Höhe** eines Baumes.

Ein Baum heisst **ausbalanciert**, wenn *für jeden* Knoten gilt, dass sich die Anzahl der Knoten in den beiden Unterbäumen höchstens um Eins unterscheiden.

- **verkettete** Datenstruktur \Rightarrow vereinfacht das Einfügen
- **nicht-lineare** Datenstruktur \Rightarrow beschleunigt die Suche
- Je flacher ein Baum, desto schneller die Algorithmen

Beispiel



Vergleich Datenstrukturen

	Daten- feld	verkettete Liste	Binär- baum
--	----------------	---------------------	----------------

Besonderheiten

statisch ⁽¹⁾	dynamisch	dynamisch
zusammen	verkettet	verkettet
linear	linear	nicht-linear

Algorithmen

unsortiert einfügen	aufwändig ⁽¹⁾	einfach	einfach
sortiert einfügen	aufwändig ⁽¹⁾	aufwändig	einfach
suche binär ⁽²⁾	schnell	-	schnell ⁽³⁾
suche sequentiell	langsam	langsam	langsam

Anmerkungen

(1) es gibt auch dynamische Datenfelder

(2) nur wenn Daten sortiert

(3) nur wenn Baum ausbalanciert

Bäume und Datenbanken

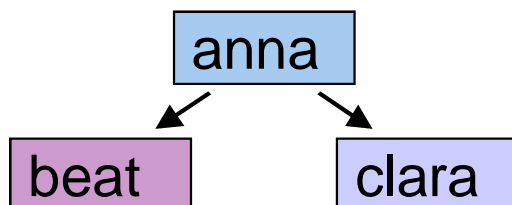
Datei

	anna	302 33 12	309	clara	267 45 21	107	beat	267 ...
--	------	-----------	-----	-------	-----------	-----	------	---------

Tabelle (Datenbank)

<i>Name</i>	<i>Telefon</i>	<i>Büro</i>
anna	302 33 12	309
clara	267 45 21	107
beat	267 45 34	108

Index



- 😊 Ein Index beschleunigt die Suche in Daten,
- 😞 verlangt jedoch zusätzlichen Speicherplatz und
- 😞 verlangsamt Schreib- und Löschoperationen.

Aufgabe

Wir möchten ein Wörterbuch 'Deutsch-Englisch' für Fachbegriffe implementieren. Später möchten wir weitere Wörterbücher erstellen (z.B. 'Englisch-Deutsch').

Benutzerschnittstelle

<i>deutsch</i>	<i>englisch</i>
<input type="text" value="baum"/>	<input type="text" value="tree"/>
<input type="button" value="füge hinzu"/>	<input type="button" value="übersetze"/> <input type="button" value="entferne"/>

<i>listeDeutsch</i>	<i>listeEnglisch</i>
ast	branch
baum	tree
blatt	leaf
kind	child
knoten	node
wurzel	root

Lösungsvarianten

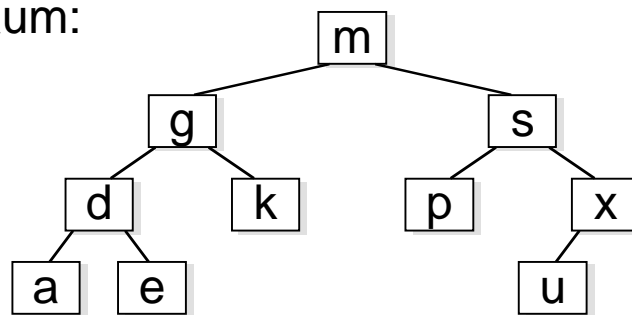
- ☹ Wörterbuch erstellen und später kopieren
- ☹ gemeinsamen Code in Prozeduren auslagern
- 😊 Wörterbuch in Klassenmodul implementieren

Baumtraversierungen

Einen Baum **traversieren** heisst ausgehend von der Wurzel, alle Knoten besuchen und jeden einmal ausgeben.

Beispiele

sortierter Binärbaum:



a) in sortierter Reihenfolge:
a, d, e, g, k, m, p, s, x, u

b) ebenenweise:
m, g, s, d, k, p, x, a, e, u

Traversierungsarten

Eine **Tiefentraversierung** besucht zunächst die Blätter und erst dann den Ebenennachbar eines Knotens.

Eine **Breitentraversierung** besucht zunächst den Ebenennachbar eines Knotens und erst dann die Blätter.

Tiefentraversierung

Verschiedene Tiefentraversierungen

Präordnungs-Traversierung

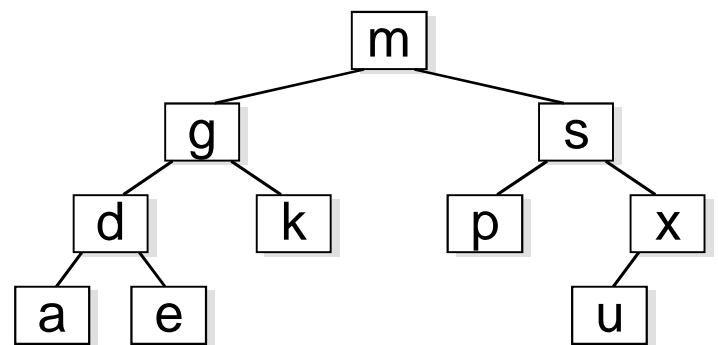
1. besuche Vater
2. besuche alle Kinder

Inordnungs-Traversierung (nur bei Binärbäumen)

1. besuche linkes Kind
2. besuche Vater
3. besuche rechtes Kind

Postordnungs-Traversierung

1. besuche alle Kinder
2. besuche Vater



Beispiele

sortierter Binärbaum:

a) Präordnungs-Traversierung:

.....

b) Inordnungs-Traversierung:

.....

c) Postordnungs-Traversierung

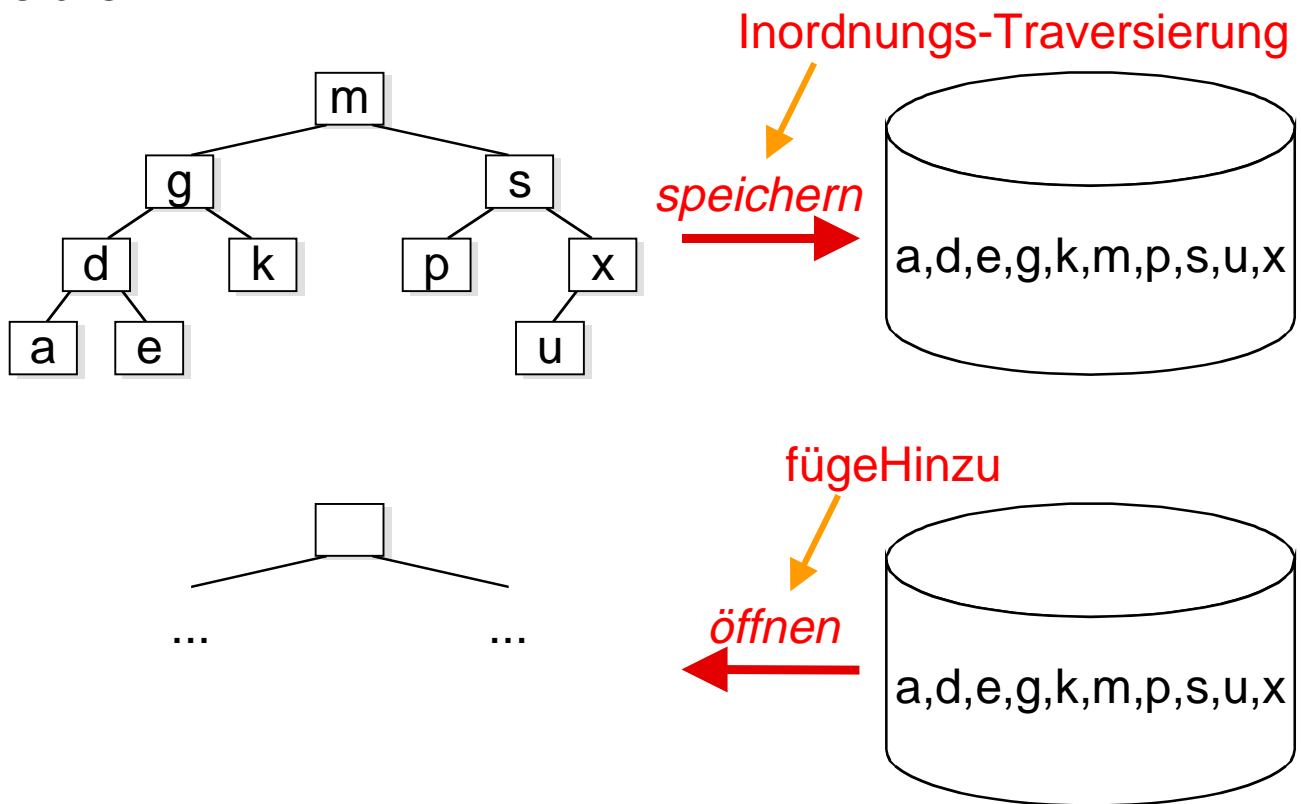


.....

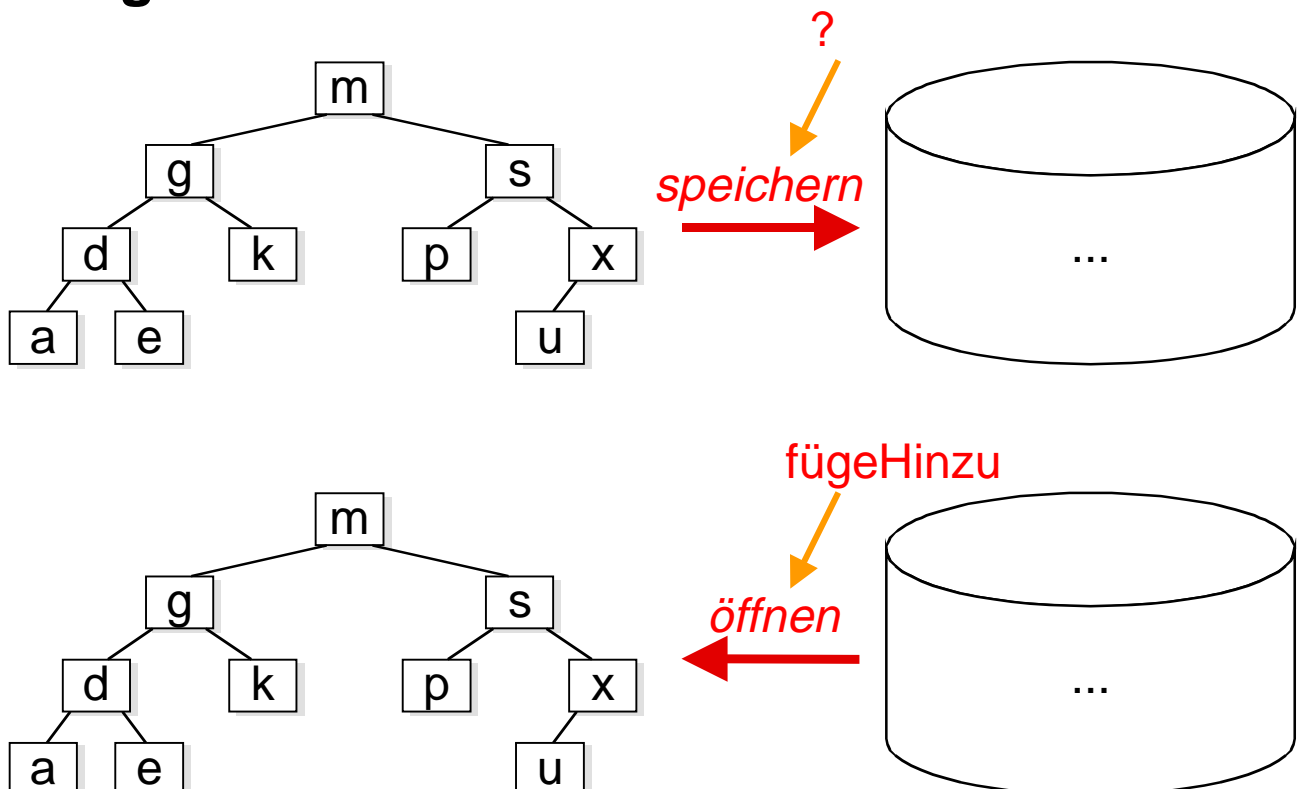
Welche Traversierung eignet sich zur Ausgabe einer nach den deutschen Begriffen (Schlüssel) sortierten Liste der Einträge unseres Wörterbuchs?

speichern

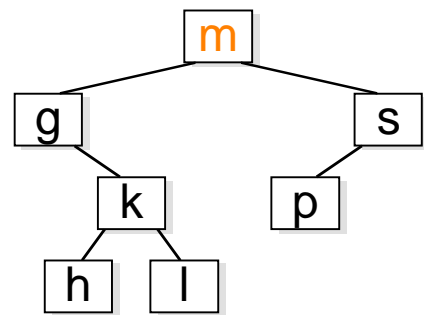
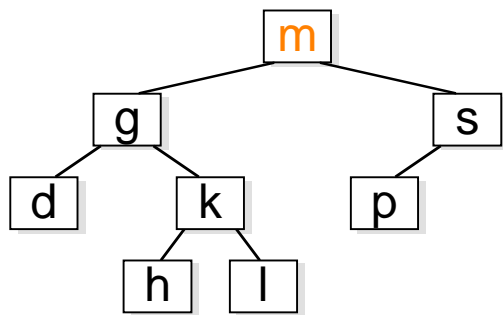
Problem



Lösung



Inordnungs-Traversierung



Entwurf OErster()

Beispiele

- OErsterRek('k')
- OErsterRek('g')
- OErsterRek('s')
- OErsterRek('d')
- OErsterRek('m')

Entwurfscod

OErsterRek(**Baum**)

'Abbruchbedingungen

FALLS **Baum** leer ist
gib Nothing zurück

SONST FALLS **Baum** KEINEN **linken** Teilbaum hat
gib **Baum** zurück

'Reduktionen

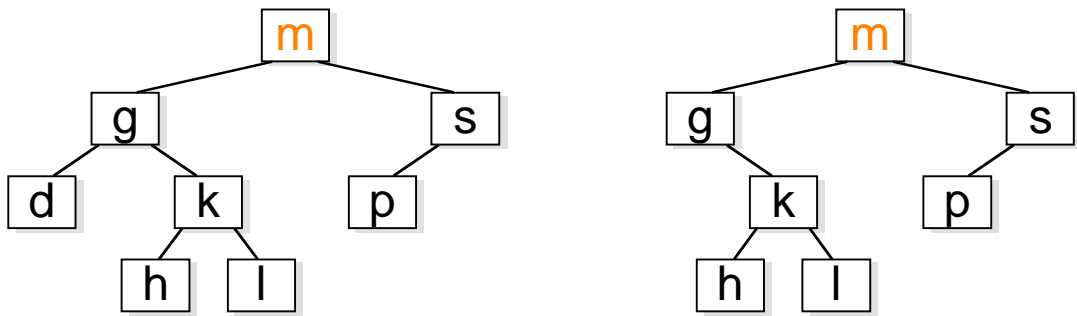
SONST

gib ersten Eintrag von **linkem** Teilbaum von **Baum** zurück

Test

⇒ Testen Sie den Algorithmus anhand der Beispiele

Inordnungs-Traversierung



Implementation OErster()

```
Public Function OErster() As CEintrag
    Set OErster = OErsterRek(Wurzel)
End Function
```

```
Private Function OErsterRek(Baum As CEintrag) _
    As CEintrag
```

```
    'Abbruchbedingungen:
```

```
If Baum Is Nothing Then
    Set OErsterRek = Nothing
```

```
ElseIf Baum.KindLi Is Nothing Then
    Set OErsterRek = Baum
```

```
    'Reduktionen
```

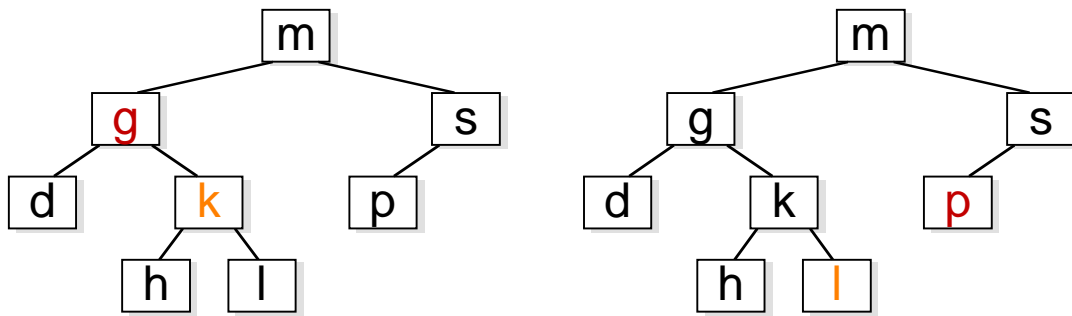
```
Else
```

```
    Set OErsterRek = OErsterRek(Baum.KindLi)
```

```
End If
```

```
End Function
```

Inordnungs-Traversierung



Entwurf ONächster()

Beispiele

- ONächster('k')
- ONächster('g')
- ONächster('l')
- ONächster('p')
- ONächster('h')

Entwurfscod

ONächster(Knoten)

FALLS Knoten einen rechten Teilbaum hat
gib ersten Eintrag im rechten Teilbaum von Knoten zurück

SONST

beginne bei Knoten

WIEDERHOLE BIS aktueller Begriff > Begriff im Knoten

klettere einen Knoten hoch

FALLS neuer aktueller Knoten leer ist

gib Nothing zurück

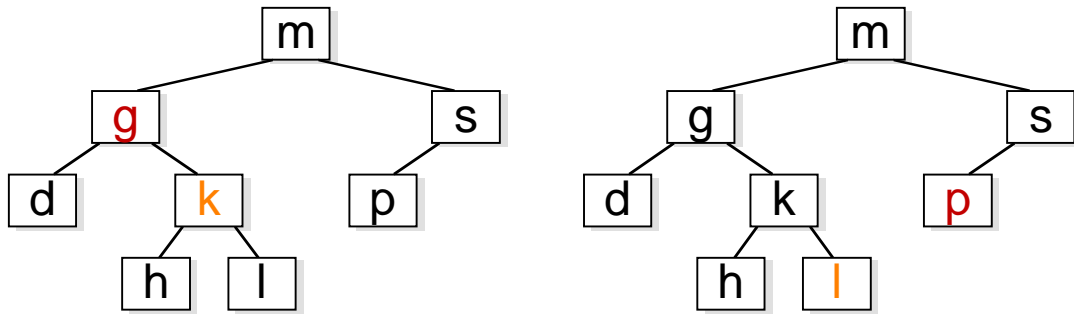
brich ab

gib aktuellen Knoten zurück

Test

Testen Sie den Algorithmus anhand der Beispiele

Inordnungs-Traversierung



Implementation ONächster()

```
Public Function ONächster(Knoten As CEintrag) _
    As CEintrag

    Dim Läufer As CEintrag
    If Not Knoten.KindRe Is Nothing Then
        Set ONächster = OErsterRek(Knoten.KindRe)
    Else
        Set Läufer = Knoten
        Do Until Läufer.Begriff > Knoten.Begriff
            Set Läufer = Läufer.Vater
            If Läufer Is Nothing Then
                Exit Do
            End If
        Loop
        Set ONächster = Läufer
    End If
End Function
```

Schnittstelle

- Collection-Objekte: For Each...Next-Schleife

Beispiel

```
For Each Tab1 In Worksheets
```

```
...
```

```
Next Tab1
```

- CWörterbuch-Objekte: *keine* For Each...Next-Schleife möglich
statt dessen zwei Funktionen:

OErster(): gibt den ersten Eintrag nach dem Inordnungs-Verfahren zurück

ONächster(Eintrag): gibt den Ordnungsnachfolger von Eintrag nach dem Inordnungs-Verfahren zurück oder Nothing, falls kein Ordnungsnachfolger existiert

Beispiel

```
Dim Knoten As CEintrag
```

```
Set Knoten = BuchDeEn.OErster()
```

```
Do While Not Knoten Is Nothing
```

```
...
```

```
Set Knoten = BuchDeEn.ONächster(Knoten)
```

```
Loop
```

Schnittstelle

(als Ersatz für For Each...Next-Schleife)

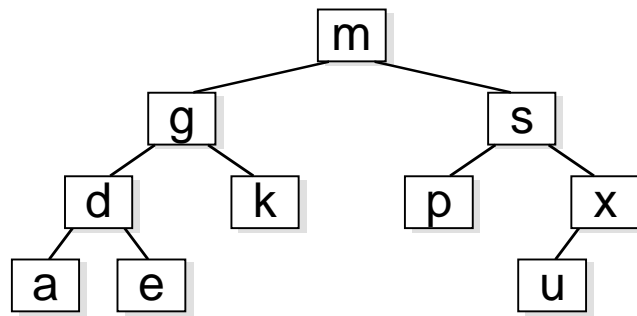
NErster(): gibt den ersten Eintrag gemäss der Breiten-traversierung zurück

NNächster(Eintrag): gibt den Nachbar von Eintrag gemäss der Breiten-traversierung zurück oder Nothing, falls kein Nachbar existiert

Aufruf

```
Dim Knoten As CEintrag
Set Knoten = NErster()
Do While Not Knoten Is Nothing
    ...
    Set Knoten = NNächster(Knoten)
Loop
```

Breitentraversierung



Entwurf NErster()

NErster()

FALLS Wurzel nicht leer ist
gib Wurzel zurück

SONST
gib Nothing zurück

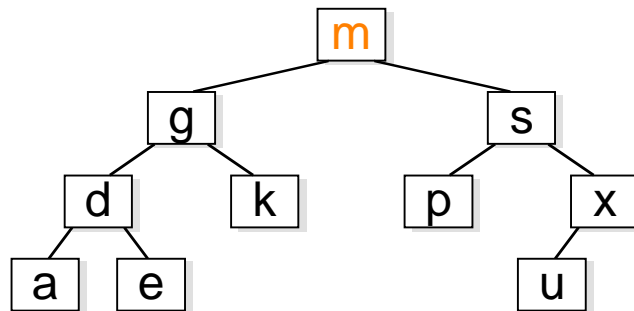
Implementation NErster()

```
Public Function NErster() As CEintrag  
    Set NErster = Wurzel  
End Function
```

Breitentraversion

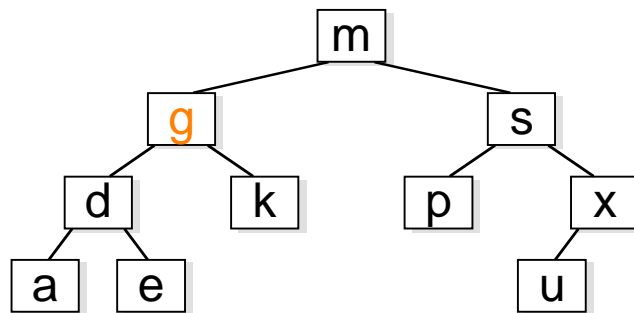
zum Entwurf von NNächster()

1.



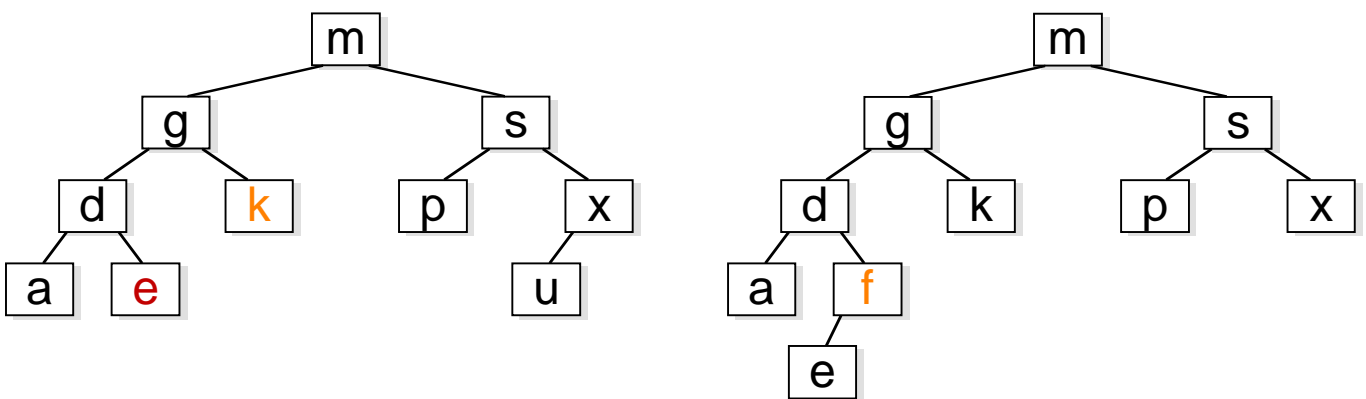
Der Nachbar der Wurzel ist ihr linkes bzw. ihr rechtes Kind. Hat die Wurzel keine Kinder, so hat sie auch keinen Nachbarn.

2.



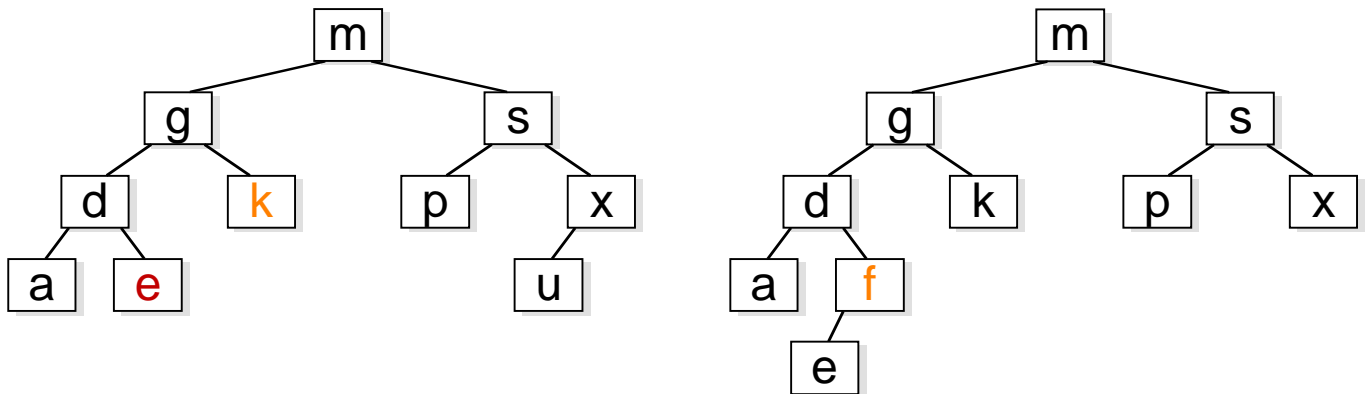
Der Nachbar eines linken Kindes mit einem rechten Bruder ist der rechte Bruder.

3.



Der Nachbar eines Kindes ohne rechten Bruder ist das linke bzw. das rechte Kind des nächsten (nicht kinderlosen) Nachbarn des Vaters.

Breitenterversierung



Entwurf NNächster()

NNächster(Knoten)

FALLS Knoten = Wurzel

FALLS Knoten einen linken Nachfolger hat
gib linken Nachfolger von Knoten zurück

SONST FALLS Knoten einen rechten Nachfolger hat
gib rechten Nachfolger von Knoten zurück

SONST
gib Nothing zurück

SONST

FALLS Knoten linker Nachfolger seines Vaters ist
UND Vater von Knoten einen rechten Nachfolger hat
gib rechten Nachfolger von Vater von Knoten zurück

SONST

Durchlaufe die Nachbarn⁽¹⁾ des Vaters von Knoten

FALLS Nachbar einen linken Nachfolger hat
gib linken Nachfolger des Nachbarn zurück

FALLS Nachbar einen rechten Nachfolger hat
gib rechten Nachfolger des Nachbarn zurück

FALLS Nachbar = Knoten

gib Nothing zurück
brich ab

⁽¹⁾ Nachbar im Sinne der Breitenterversierung \Rightarrow Rekursion

Breitentersersierung

Implementation NNächster()

```
Public Function NNächster(Knoten As CEintrag) _
    As CEintrag
    Dim Läufer As CEintrag
    If Knoten Is Wurzel Then
        If Not Knoten.KindLi Is Nothing Then
            Set NNächster = Knoten.KindLi
        ElseIf Not Knoten.KindRe Is Nothing Then
            Set NNächster = Knoten.KindRe
        Else
            Set NNächster = Nothing
        End If
    Else
        If Knoten Is Knoten.Vater.KindLi _
            And Not Knoten.Vater.KindRe Is Nothing Then
            Set NNächster = Knoten.Vater.KindRe
        Else
            Set Läufer = Knoten.Vater
            Do While Not Läufer Is Knoten
                Set Läufer = NNächster(Läufer)
                If Not Läufer.KindLi Is Nothing Then
                    Set NNächster = Läufer.KindLi
                    Exit Do
                ElseIf Not Läufer.KindRe Is Nothing Then
                    Set NNächster = Läufer.KindRe
                    Exit Do
                End If
            Loop
        End If
    End If
End Function
```