Syntax

1. Aufgabe :

Geben Sie zu jeder der folgenden einfachen Sprachen jeweils eine Grammatik an, mit der alle Worte dieser Sprache erzeugt werden können.

(a) Worte beliebiger Länge, die nur aus dem Buchstaben ’a’ bestehen.

(b) Alle symmetrischen Worte aus den Buchstaben ’a’ und ’b’ (also z.B. ’a’, ’aba’, ’bbbb’, ’abaaba’).

(c) Eine beliebige Folge von Namen (Vorname und Nachname) Ihrer Übungsgruppe durch Komma getrennt. Dabei sollen die Vornamen optional sein.

2. Aufgabe :

(a) Entwickeln Sie eine Grammatik in EBNF-Notation für Literaturangaben der Form:

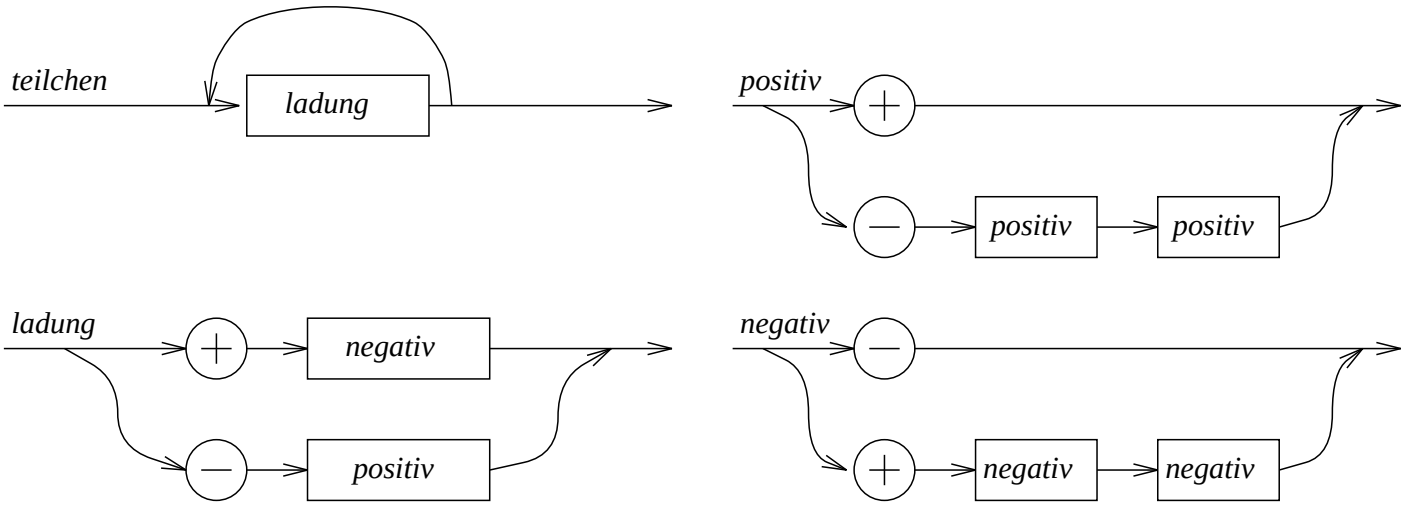
```
Autor/inn/en:  Titel, Verlag, Erscheinungsjahr, Seitenzahl,
referenzierte Seite(n).
```

Mit referenzierte Seite(n) ist / sind die Seiten der Veröffentlichung gemeint, auf die sich die Literaturangabe bezieht. Die Nichtterminale für Wort bzw. Zahl brauchen nicht verfeinert werden!

(b) Übersetzen Sie die EBNF-Regeln aus Teil a) in Syntaxdiagramme. item Können Sie mit Hilfe von EBNF-Regeln oder Syntaxdiagrammen angeben, daß die referenzierte(n) Seite(n) innerhalb der angegebenen Seitenzahl liegen müssen? Wenn ja wie? Wenn nein, warum nicht?

3. Aufgabe :

Gegeben seien die folgenden Synatxdiagramme:



(a) Was für Worte kann man mit diesen Regeln erzeugen (das Startsymbol ist $teilchen$)?

(b) Übersetzen Sie die Diagramme in EBNF-Regeln. item Finden Sie eine Grammatik (in EBNF-Notation oder Syntaxdiagrammen), mit der die gleichen Worte erzeugt werden, die aber weniger Regeln hat.

4. Aufgabe :

Ein Programm in C++ besteht aus unterschiedlichen Komponenten, die die verschiedenen Aspekte eines Programms widerspiegeln. Prinzipiell kann man vier Arten von Bausteinen unterscheiden, aus denen sich ein Programm zusammensetzt.

Grobe Programmstruktur: Hierunter fallen die Aufteilung eines Programms in Prozeduren und Funktionen sowie, insbesondere bei großen Programmen, die Einteilung in Module.

Deklarationen: Alles, was in den Anweisungen eines Programms verwendet wird, muß zuvor deklariert, also bekannt gemacht werden.

Anweisungen: Die Teile eines Programms, die bei seiner Ausführung bestimmen, was geschieht.

Ausdrücke: Will man etwas über die Werte der Variablen eines Programms wissen (z.B. in der Bedingung einer IF-Anweisung) oder deren Werte ändern, so benötigt man Ausdrücke, um mit diesen Werten umzugehen.

In der Syntax einer Programmiersprachen müssen diese Bestandteile der Sprache definiert werden. Betrachten Sie die Grammatik von C++ im Anhang I Skript.

(a) Schreiben Sie für jeden der vier beschriebenen Bereiche 5 Regeln aus der Grammatik auf, die Sie diesem Bereich zuordnen würden.

(b) Können Sie jede Regel eindeutig einem der Bereiche zuordnen? Bei welchen Regeln scheint das nicht möglich zu sein und warum (drei Beispiele mit Begründung).

5. Aufgabe :

- Welche der folgenden C-Programme sind korrekt und was geben Sie (abhängig vom eingelesenen Wert von a) aus?

1.
```
#include <iostream.h>
void main(void){
char a;
cin >> a;
if (a=='w'){
  cout << "Hallo";
  cout << " Welt!\n";}
else{
  cout << "Hallo ";
  cout << " Erde!\n";}
}
```

2.
```
#include <iostream.h>
void main(void){
char a;
cin >> a;
if (a=='w')
  printf("Hallo");
  cout << " Welt!\n";
else
  cout << "Hallo";
  cout << " Erde!\n";
}
```

3.
```
#include <iostream.h>
void main(void){
char a;
cin >> a;
if (a=='w'){
  cout << "Hallo";
  cout << " Welt!\n";}
else
  cout << "Hallo";
  cout << " Erde!\n";
}
```

4.
```
#include <iostream.h>
void main(void){
char a;
cin >> a;
cout << "Hallo";
if (a=='w')
  cout << " Welt!\n";
else
  cout << " Erde!\n";
}
```

5.
```
#include <iostream.h>
void main(void){
char a;
cin >> a;
cout << "Hallo";
switch(a){
  case 'w':
    cout << " Welt!\n";
  default:
    cout << " Erde!\n";
  }
}
```

6.
```
#include <iostream.h>
void main(void){
char a;
cin >> a;
cout << "Hallo";
switch(a){
  case 'w':
    cout << " Welt!\n";
    break;
  default:
    cout << " Erde!\n";
  }
}
```

- Was geben die folgenden Anweisungen aus (i sei als int i deklariert)?

1. for(i = 0; i < 5; i++) cout « i « " ";
2. for(i = 0; i < 5; i++) {cout « i « " ";}
3. for(i = 0; i < 5; i++); cout « i « " ";
4. for(i = 0; i < 5;) cout « i++ « " ";
5. for(i = 0; i < 5;) cout « ++i « " ";
6. for(i = 0; i < 5;) cout « (++i -1) « " ";

6. Aufgabe :

- In C(++) gibt es die Möglichkeit auch komplexe Datenstrukturen mit Hilfe von Literalen zu initialisieren. Deklarieren Sie ein Dreidimensionales Feld von ganzen Zahlen wuerfel, wobei jede Dimension die Größe 4 hat (also ein Würfel mit Kantenlänge 4). Initialisieren Sie dieses Feld mit beliebigen Werten.

- Für Zeichenketten gibt es eine Schreiberleichterung. Folgende Deklaration ist gültig:
  `char name[80] = "Arthur".`
  Deklarieren Sie ein Zweidimensionales Feld von Zeichen (5×80) und initialisieren Sie es mit verschiedenen Namen.
- Definieren Sie einen Verbundtyp Person, der Komponenten für Namen, Vornamen und Alter einer Person vorsieht. Deklarieren Sie eine Variable dieses Typs und weisen Sie ihr einen Wert zu.

7. Aufgabe :

Schreiben Sie ein Programm in C++, das den Benutzer nach einem Datum fragt, es einliest und berechnet, wieviele Tage es noch bis zum nächsten Weihnachtsfest sind. Realisieren Sie dazu eine Funktion DatumNachJahrestag, die ein übergebenes Datum in die Anzahl der im Jahr bisher vergangenen Tage umrechnet. Die Funktion und das Hauptprogramm sollen in getrennten Dateien gespeichert werden, das Hauptprogramm muß also mit Hilfe einer Header-Datei diese Funktion importieren (Funktionsmodul).

8. Aufgabe :

In dieser Aufgabe sollen Sie einen Datentyp entwerfen, mit dem Sie Ihren Wochenstundenplan speichern können. Dazu benötigen Sie verschiedene Hilfsmittel:

(a) Deklarieren Sie einen geeigneten Aufzählungstyp Wochentag.

(b) Deklarieren Sie einen Verbundtyp Veranstaltung, der den Inhalt und die Art einer Veranstaltung aufnehmen kann. Ein solcher Verbund soll den Titel der Veranstaltung, den Namen des Dozenten, die Art der Veranstaltung (Vorlesung oder Übung) und den Veranstaltungsort aufnehmen können.

(c) Deklarieren Sie einen zweidimensionalen Feldtyp Stundenplan, dessen erste Dimension den Wochentag und dessen zweite Dimension die Uhrzeit einer Veranstaltung darstellt. Die Komponenten des Feldtyps haben den Typ Veranstaltung. Zur Vereinfachung können Sie davon ausgehen, daß Veranstaltungen nur zur vollen Stunde beginnen und eine Stunde lang dauern.

(d) Weisen Sie einer Variable vom Typ Stundenplan die Werte für Ihre Kleingruppenübung zu (mit gerundeten Anfangs- und Endzeiten).

(e) Wie können die Vereinfachungen, die in Teil c) getroffen wurde aufgehoben werden? Bedenken Sie, daß Veranstaltungen nicht nur jeweils viertelstündlich beginnen können sondern auch unterschiedliche Längen haben. Wie sieht die Zuweisung aus Teil d) jetzt aus?

9. Aufgabe *(Rechneraufgabe)*:

Die vereinigten Staaten von Alphanumerica haben im Zuge der Automatisierung ihrer Flaggenindustrie einen Wettbewerb für die eleganteste Programmierung ihrer Flagge ausgeschrieben. Die Flagge hat folgendes Aussehen:

```
________**********
____********____********
__-****__-****__-****__-****
_**_**_**_**_**_**_**_**
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

1. Zeile: k Minus- gefolgt von k Sternsymbolen
2. Zeile: zweimal (k / 2 Minus, k / 2, Sterne)
3. Zeile: viermal (k / 4 Minus, k / 4, Sterne)
4. Zeile: achtmal (k / 8 Minus, k / 8, Sterne)

usw.

Die Flagge gibt es in verschiedenen Größen für k$\in$\{2, 4, 8, 16, 32\} (im Beispiel ist k = 16).

(a) Schreiben Sie ein C++-Programm, das für eine eingelesene Zahl k aus obiger Menge die entsprechende Flagge ausgibt. Das Programm sollte eine **rekursive** Prozedur SchreibeFlagge verwenden, die das Zeichnen einer Flagge übernimmt. Wie sieht die Schnittstelle der Prozedur aus?

(b) Ändern Sie das Programm so ab, daß nun eine nicht-rekursive Prozedur die Flagge ausgibt. Inwiefern abstrahiert die Verwendung einer Prozedur beim Zeichnen der Flagge von der gewählten Lösung?

10. Aufgabe *(Rechneraufgabe)*:

Eine natürliche Zahl $n \in \mathbb{N}$ heißt Primzahl, falls sie nur durch 1 und sich selbst ohne Rest teilbar ist. Ist n keine Primzahl, so gibt es eine Zahl $k \in \mathbb{N}$ mit $k^2 \leq n$, durch die n ohne Rest teilbar ist.

Schreiben Sie ein Programm, das alle Primzahlen ausgibt, die kleiner einer einzulesenden Zahl $m \in \mathbb{N}$ sind. Gestalten Sie das Programm so, daß es leicht lesbar und verständlich ist. Also: Kommentieren Sie ausführlich, und setzen Sie die einzelnen Programmteile textuell voneinander ab (Einrücken).

Hinweis: Zur Lösung können Sie das „Sieb des Erathostenes“ verwenden:

- Schreiben Sie alle Zahlen kleiner m auf, und streichen Sie die 1;
- Nehmen Sie die kleinste noch nicht gestrichene Zahl k, und streichen Sie alle Vielfachen davon (da sie durch k teilbar sind, können sie keine Primzahlen sein);
- Wiederholen Sie den letzten Schritt bis $k^2 > m$ gilt.

11. Aufgabe *(Rechneraufgabe)*:

In dieser Aufgabe sollen zwei Programme entwickelt werden, die das arithmetische Mittel einer Folge von reellen Zahlen in unterschiedlicher Weise berechnen. Die erste Programmvariante speichert die gelesene Zahlenfolge und berechnet nach Eingabe der letzten Zahl einmal das arithmetische Mittel und gibt das Ergebnis aus. Die Länge der Zahlenfolge soll beliebig sein, **die Speicherung der Folge in einer Array-Variablen scheidet also für die Implementierung aus**. Stattdessen soll eine dynamische Datenstruktur (linear verkettete Liste) verwendet werden.

In der zweiten Variante werden die Zahlen nicht gespeichert. Vielmehr merkt sich das Programm nur ein Zwischenergebnis und berechnet nach Eingabe jeder neuen Zahl den Mittelwert der erweiterten Folge (dieser Wert kann dann auch nach jeder Eingabe ausgegeben werden).

Zur Lösung dieser Aufgabe sollten Sie folgende Punkte beachten:

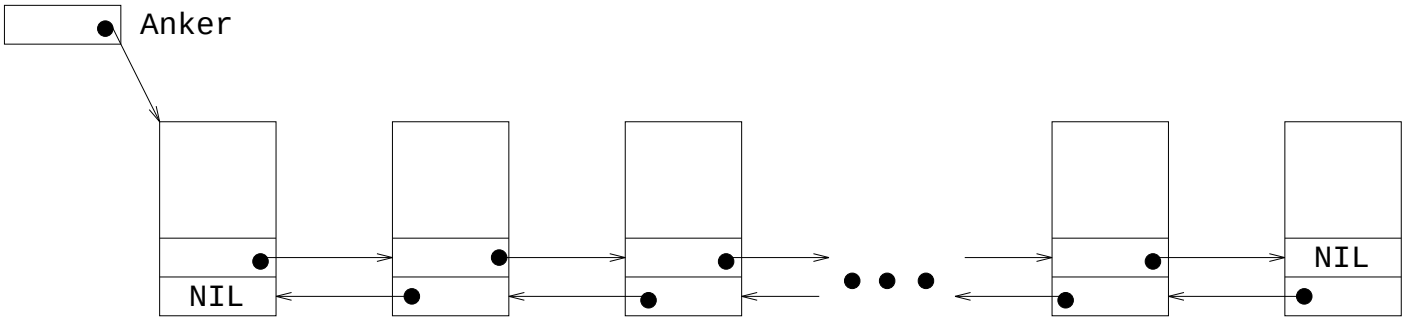
- Überlegen Sie sich einen geeigneten Dialog zwischen Programm und Benutzer. (Insbesondere: wie signalisiert der Benutzer das Ende der Zahlenfolge?)
- Entwerfen Sie einen Datentyp, mit dem die verkettete Liste der ersten Programmvariante realisiert werden kann.
- Wie kann das Zwischenergebnis der zweiten Variante am günstigsten gespeichert werden? Da das Rechnen mit Fließkommazahlen (z.B. den float,double-Zahlen aus C++) immer Ungenauigkeiten mit sich bringt, sollten so wenig Rechenschritte wie möglich gemacht werden.

Welche Variante würden Sie für diese Aufgabe bevorzugen und warum? Welche Variante ist günstiger, wenn z.B. zusätzlich eine grafische Aufbereitung der Zahlenfolge (Meßwerte darstellen) erfolgen soll?

12. Aufgabe *(Rechneraufgabe)*:

In manchen Anwendungen ist es wünschenswert, jeden Eintrag in einer Liste nicht nur mit dem nachfolgenden

Element zu verketten (einfach verkettete Liste), sondern ebenfalls mit dem vorangehenden Element. Dadurch wird beispielsweise das Löschen eines Elements aus der Liste wesentlich vereinfacht. Man hat dann eine *doppelt verkettete Liste*.



Schreiben Sie ein C++ Programm, das eine doppelt verkettete Liste realisiert. Es genügt, wenn sie außer dem Datentyp die Operationen `Einfuegen` und `Streichen` implementieren (`Einfuegen`: z.B. Einfügen vor einem bestimmten Element in der Liste; `Streichen`: beliebiges Element aus der Liste entfernen). Überlegen Sie sich, wie die Operationen im allgemeinen Fall funktionieren und welche Sonderfälle zu beachten sind.

Sortieren

13. Aufgabe *(Rechneraufgabe)*:

Schreiben Sie eine Prozedur, die eine gegebene doppelt verkettete Liste von Integer/int-Zahlen in der folgenden Weise sortiert: Die Liste wird von Anfang in Richtung Ende einmal durchlaufen und dabei wird jedes Element, das kleiner als das vorangegangene aktuelle Element ist, an seinen richtigen Platz weiter vorne in der Liste einsortiert. Dieses Verfahren wird „Sortieren durch Einfügen“ oder „Straight Insertion“ genannt.

Beispiel: Anfang

12, 3, 7, 19, 10.

1. Schritt: 3 ist das erste Element, das kleiner als sein Vorgänger in der Liste ist, also muß es nach vorne verschoben werden; als Ergebnis erhält man:

   3, 12, 7, 19, 10.

   Hierbei merkt man sich das nächste noch nicht geprüfte Element, also die 7.

2. Schritt: 7 ist wieder kleiner als 12, an der richtigen Stelle einfügen ergibt dann

   3, 7, 12, 19, 10.

3. Schritt: Da 19 korrekt steht, muß nur noch 10 umsortiert werden:

   3, 7, 10, 12, 19.

14. Aufgabe :

- Schreiben Sie eine Prozedur *Mergesort*, die eine (einfach oder doppelt) verkette Liste von int-Zahlen in der folgenden Weise sortiert: Zunächst wird die Liste in zwei möglichst gleich große Listen geteilt. Jede der beiden Listen wird nun für sich alleine durch *Mergesort* sortiert. Die beiden sortierten Listen werden nun wieder nach und nach zu einer einzigen zusammengefügt (*merge*) indem man ihre jeweils ersten Elemente miteinander vergleicht und das kleinere in die zusammengesetzte Liste einfügt.

- Schreiben Sie zum Testen der Prozedur ein Programm, das eine Folge von Zahlen einliest, mit der Prozedur Mergesort sortiert und das Ergebnis schließlich wieder ausgibt.

  Skizzieren Sie die Vorgänge beim „Sortieren durch Mischen“ (*Mergesort*) anhand der folgenden Zahlenfolge, indem Sie die internen Zwischenzustände der Zahlenfolge in geeigneter Form darstellen:

  15, 22, 7, 14, 48, 35, 20, 29, 9, 18, 38, 51, 27, 11, 31.

15. Aufgabe $\star$:

Ein elegantes und auch effizientes Sortierverfahren ist Heapsort. Heapsort arbeitet nach dem Prinzip Sortieren durch Auswahl. Der Unterschied zu „Straight Selection“ besteht darin, daß in dem Teil des Feldes, der noch unsortiert ist, eine Struktur aufgebaut wird, die das Auffinden des größten noch nicht sortierten Elementes erheblich beschleunigt.

Ein Heap (Haufen) hat eine baumartige Struktur. Ein Element in einem Heap hat i.a. zwei Nachfolger. Sei A ein Feld mit N Elementen. Die Nachfolger des Elementes mit Index $1 \leq i \leq N/2$ sind $A[2\star i]$ und $A[2\star i + 1]$ (falls $i = N/2$ und N gerade ist, dann hat $A[i]$ nur einen Nachfolger). Alle Elemente mit Index $i > N/2$ haben keine Nachfolger. Sie werden die Blätter des Baumes genannt. Das Element mit Index 1 nennen wir Wurzel. Ein Beispiel für die Abbildung eines Feldes in solch eine Baumstruktur zeigt Abb. 1. Ein Baum, wie er in der Abbildung zu sehen ist, ist aber nur ein Modell, mit dem man sich die Struktur des Heaps besser veranschaulichen kann. Gespeichert wird er in einer Feldvariablen!

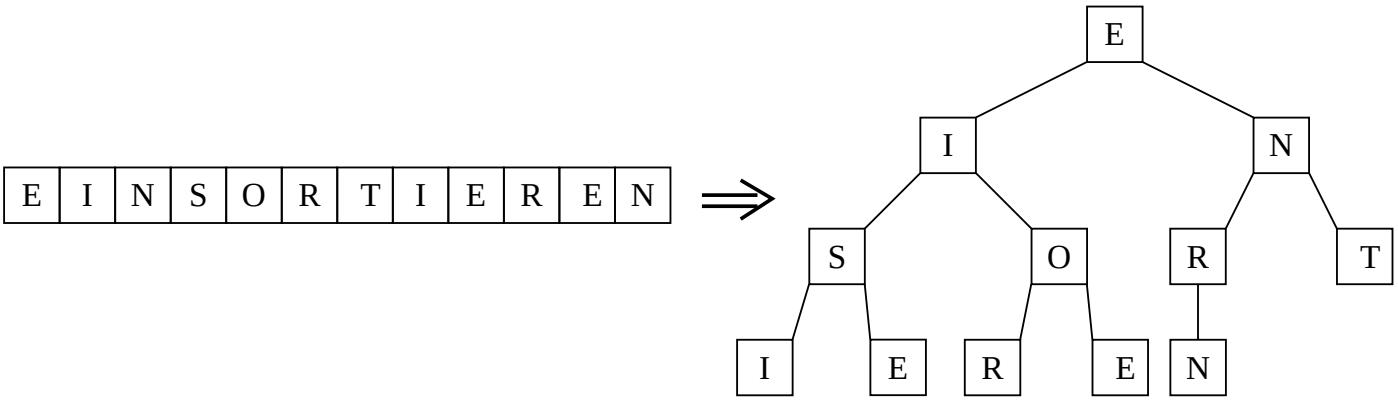


Abbildung 1: Baumartige Anordnung eines Beispielfeldes.

Das dargestellte Feld ist allerdings noch kein Heap. In einem Heap ist jedes Element des Feldes größer als seine Nachfolger. Man kann ein Feld durch Vertauschoperationen in einen Heap überführen. Für das obige Beispiel sieht das Feld bzw. der Baum nach diesen Vertauschungen wie in Abb. 2 aus.

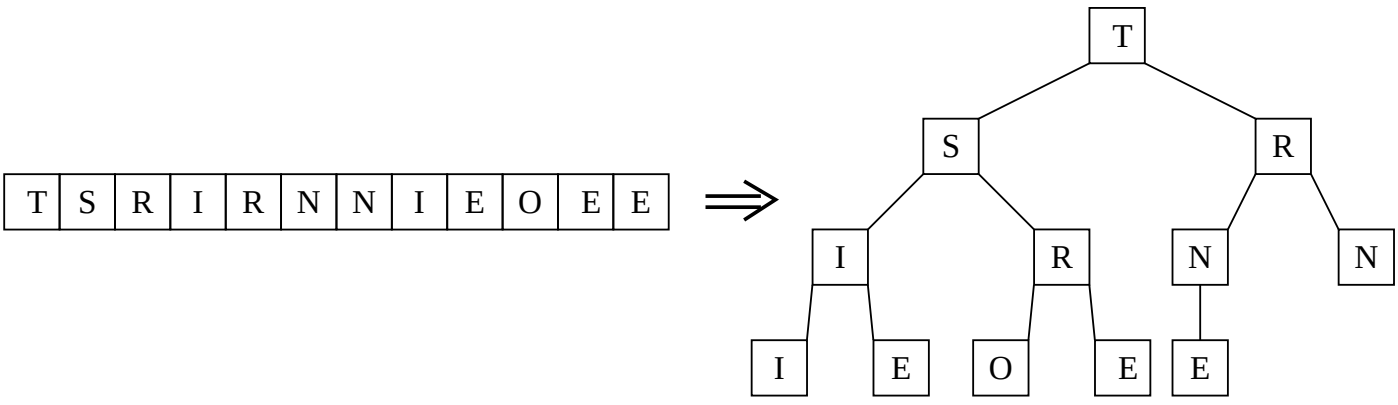


Abbildung 2: Zum Heap umsortiertes Beispielfeld.

(a) Um ein Feld in einen Heap zu überführen, kann man folgerndermaßen vorgehen. Die Elemente mit den Indizes `N / 2 + 1` bis `N` haben keine Nachfolger. Sie sind also bereits Heaps. Nun geht man rückwärts von `N / 2` bis 1 und betrachtet die entsprechenden Elemente als Wurzeln von Unterbäumen. In diesen Bäumen muß nur noch untersucht werden, ob die Wurzel an der richtigen Stelle im Baum ist, damit dieser (Unter-) Baum einen (Unter-) Heap bildet. Ist also einer der Nachfolger des untersuchten Feldelementes (der Wurzel des Unterbaumes) größer als die Wurzel, so vertauscht man die Wurzel mit dem größeren der beiden Nachfolger. Dieses Vertauschen muß solange fortgesetzt werden bis das ursprüngliche Wurzelelement nur Nachfolger hat, die kleiner oder gleich groß sind, wie es selbst. Schreiben Sie eine Prozedur `HeapAbwaerts`, die diese Vertauschungen für einen Unterbaum (gekennzeichnet durch einen Feldindex) vornimmt.

(b) Das eigentliche Sortieren ist nun sehr einfach. Durch die Heapstruktur ist sichergestell, daß das Element mit Index 1 (die Wurzel des Baumes) das größte Element des Feldes ist. Man vertauscht die Wurzel mit dem letzten Element des Feldes, das noch zum Heap gehört. Danach ist die Größe des Heaps um 1 verringert und die Heapeigenschaft muß wiederhergestellt werden, weil das neue Wurzelelement i.a. kleiner als seine Nachfolger sein wird. Diese Schritte werden solange wiederholt, bis der verbleibende Heap die Größe 1 hat. Implementieren Sie dieses Sortierverfahren (dazu gehört auch, vor dem eigentlichen Sortieren, das Feld in einen Heap zu überführen).

(c) Schätzen Sie ab, wieviele Vergleiche im Mittel beim Sortieren mit Heapsort durchgeführt werden müssen. Beachten Sie, daß auch das erstmalige Überführen des Feldes in einen Heap berücksichtigt werden muß.

16. Aufgabe :

Eine Anwendung der Sortierung von Feldern sind Suchalgorithmen. Um einen bestimmten Schlüssel in einem unsortierten Feld der Länge N zu finden, benötigt man im Mittel $\frac{N}{2}$ Vergleiche. Wesentlich bessere Ergebnisse kann man auf sortierten Feldern erzielen. Eines der effizientesten Verfahren ist die binäre Suche.

Gegeben sei ein Feld `A`. In `A` sucht man zwischen den Indizes `l` und `h` (`l` $<$ `h`) den Schlüssel `s`. Dazu betrachtet man zuerst das Element mit dem Index $i = (l + h) / 2$. Gilt `A[i]` $=$ `s`, so endet die Suche. Ist `A[i]` $>$ `s`, so wiederholt man die Suche in den Indexgrenzen `l` bis `i` -1, sonst mit `i` $+1$ bis `h`.

(a) Gegeben sei ein Feld `A` mit 100 Elementen vom Typ `int`, mit `A[i]` $=$ `i` für alle `i` von 1 bis 100. Welche vergleiche müssen bei der binären Suche nach dem Schlüssel 35 durchgeführt werden?

(b) Implementieren Sie die binäre Suche auf Feldern mit ganzzahligen Komponenten. Achten Sie besonders auf Terminationsbedingungen und Indexgrenzen (was passiert, wenn der gesuchte Schlüssel nicht im Feld ist).

(c) Zu jedem Programm gehört eine Dokumentation. Während die Verwendung von Kommentaren inzwischen selbstverständlich sein sollte (auch für das Programm aus Teil a), ist weitergehende Dokumentation bei kleinen Programmen eher selten zu finden. Gerade für kompakte Algorithmen erleichtert die zusätzliche Dokumentation aber das Verständnis. Zeichnen Sie daher zu dem in a) entwickelten Programmteil, der die binäre Suche realisiert, ein Struktogramm.

(d) Schätzen Sie ab, wieviele Vergleich im schlechtesten Fall nötig sind, damit die binäre Suche (erfolgreich oder nicht erfolgreich) terminiert. Machen Sie sich den Unterschied zur Suche in einem unsortierten Feld anhand eines Zahlenbeispiels (10000 Elemente in einem Feld) klar.

Testen

17. Aufgabe :

Gegeben sei ein Programm, das drei ganze Zahlen einliest und sie als Längen der Seiten eines Dreiecks interpretiert. Das Programm druckt eine Meldung mit der Feststellung aus, ob das Dreieck ungleichseitig, gleichschenklig oder gleichseitig ist.

Zu diesem Programm sollen Sie einen Blackbox-Test durchführen. Denken Sie sich dazu sinnvolle Testfälle aus und notieren Sie mit Kommentaren, wozu sie dienen sollen.

18. Aufgabe :

Folgender Algorithmus sortiert ein Feld `a` der Länge `n` in aufsteigender Reihenfolge mittels „Sortieren durch Einfügen“. Stellen Sie zur Durchführung eines Whitebox-Tests den Flußgraphen für den Algorithmus auf, und geben Sie geeignete Testfälle an, die (zusammengenommen) alle Pfade des Flußgraphen durchlaufen.

```
for (int i=1; i<n; i++) {
   x = a[i]; L = 0; R = i;

   while (L<R) {
      m = (L+R)/2;
      if (a[m] <= x) {
         L = m+1; };
      else {
         R = m;
      };
   };

   // Einfuegestelle (R) gefunden
   for (int j=i; j<=R+1; j--) {
      a[j] = a[j-1];
   };
   a[R] = x;
};
```

19. Aufgabe :

Stellen Sie zur Durchführung eines Whitebox-Tests den Flußgraphen für den in Aufgabe 21 implementierten Algorithmus der binären Suche auf, und geben Sie geeignete Testfälle an, die (zusammengenommen) alle Pfade des Flußgraphen durchlaufen.

Module und Datenabstraktion

20. Aufgabe :

In der Vorlesung sind die Schnittstelle und der Rumpf eines abstrakten Datenobjekts „Buffer“ (Synonyme: Puffer, Schlange, Queue) vorgestellt worden. Ein Puffer ist eine sequentielle Datenstruktur, bei der Elemente an einem Ende eingefügt und am anderen Ende ausgelesen werden (FIFO: first in first out). Bei einem „Keller“ (Synonyme: Stapel, Stack) hingegen werden die Datenelemente am gleichen Ende eingefügt und wieder entnommen (LIFO: last in first out).

- Wie müßte die Schnittstelle eines abstrakten Datenobjekts „Keller“ aussehen? Definieren Sie diese Schnittstelle in C++ und implementieren Sie den Rumpf dazu.

- Definieren Sie die Schnittstelle eines abstrakten Datentyps „Keller“. Was ist anders als beim abstrakten Datenobjekt.

Datenstruktur Liste

21. Aufgabe :

In der Vorlesung haben Sie als Realisierungstechniken für Listen unter anderem die *nicht zirkuläre sequentielle Realisierung*, die *einfach verkettete Zeigerrealisierung* und die *einfach verkettete Indexrealisierung* kennengelernt.

(a) Schreiben Sie für jede der genannten Techniken eine Prozedur, die ein Element nach dem aktuellen Element in eine übergebene Liste einfügt (Typdefinitionen wie im Skript):

```
void ElementEinfuegen(Datenstr Liste, text Element);
```

(b) Diskutieren Sie die Vor- und Nachteile der genannten Realisierungstechniken.

22. Aufgabe $\star$:

In dieser Aufgabe sollen Sie einen abstrakten Datentypmodul `Liste` implementieren, der eine lineare Liste verkapselt. Die Liste soll Elemente vom Typ `ElementTyp` (von Ihnen zu wählen) speichern. Der Modul soll folgende Operationen zur Verfügung stellen:

- Erzeugen und Löschen von Listen,
- Abfragen, ob eine Liste leer oder voll ist,
- Einfügen von Elementen in eine Liste, Löschen von Elementen aus einer Liste (am Ende, am Anfang der Liste, vor oder nach dem aktuellen Element (s.u.))
- Abfragen, ob ein Element in einer Liste vorhanden ist

Auf die Elemente der Listen soll nur über ihren Wert zugegriffen werden können (es werden also keine Zeiger / Indizes von Elementen über die Schnittstelle gereicht). Um trotzdem Elemente der Listen an bestimmten Positionen löschen / einfügen zu können, soll zu jeder Liste ein „aktuelles“ Element existieren, auf das sich Operationen wie Löschen und Einfügen beziehen. Dazu benötigt man weitere Operationen:

- Gehe zum Anfang / Ende einer Liste
- Gehe zum nächsten / vorherigen Element der Liste

(a) Schreiben Sie die Schnittstelle des Moduls in C++. Kommentieren Sie die Schnittstellenoperationen ausführlich.

(b) Implementieren Sie den Rumpf des Moduls. Dabei können Sie zwischen verschiedenen Realisierungen von Listen wählen: Felder, einfach oder doppelt verkettete Listen. Achten Sie darauf, daß zu jeder Liste neben der reinen Listeninformation auch noch das aktuelle Element verwaltet werden muß.

(c) Implementieren Sie einen Modul (das Hauptprogramm), daß den Modul `Liste` benutzt. Das Programm soll eine kleine Liste aufbauen, die dann mit dem Verfahren „Straight Selection“ sortiert wird.

(d) Was ändert sich an der Schnittstelle des Moduls, wenn statt eines abstrakten Datentyps ein abstraktes Datenobjekt realisiert werden soll? Was muss geändert werden, wenn ein anderer Datentyp als der von Ihnen gewählte gespeichert werden soll?

Datenstruktur Baum

23. Aufgabe *Allgemeiner Baum*:

(a) Berechnen Sie zum Binärbaum aus Abbildung 3 Preorder-, Postorder und Inorder-Durchläufe.

(b) Gegeben Sei die nebenstehende Schnittstelle eines ADT Baum. Schreiben Sie eine rekursive C++-Prozedur `Postorder`, die durch einen Baum (vom Typ `Tree`) einen Postorder-Durchlauf macht und die Markierungen (Label) seiner Knoten dabei ausgibt.

(c) Bei Binärbäumen mit eindeutigen Markierungen ist es möglich aus gegebenen Preorder- und Postorder-Durchläufen den ursprünglichen Baum zu rekonstruieren. Geben Sie einen Algorithmus an (informell), der dies leistet und wenden Sie ihn auf das folgende Beispiel an.

Preorder:	10	5	3	16	12	20	18	23
Postorder:	3	5	12	18	23	20	16	10

Die Rekonstruktion ist nicht immer eindeutig möglich. Wann gibt es bei der Rekonstruktion Wahlmöglichkeiten und welche Information braucht man, um die Rekonstruktion eindeutig zu machen?

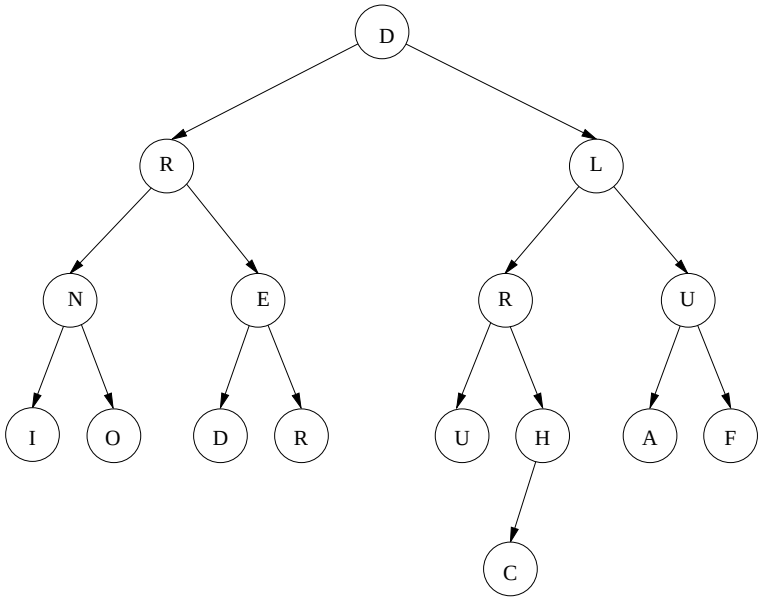


Abbildung 3: Binärbaum

24. Aufgabe *Balancierte Bäume*:

Gegeben sei folgende Typdeklaration zum Aufbau der Teile eines Binärbaums:

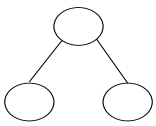
```
class BinBaum {
```

```
class ADTTree {
public:
   typedef unsigned int NodeIdT; // Jeder Knoten besitzt eine eindeutige Nummer

   ADTTree();

   void makeEmpty();
   bool isEmpty();

   NodeIdT root();

   NodeIdT leftmostChild(NodeIdT node);
   NodeIdT rightSibling(NodeIdT node);
   unsigned int label(NodeIdT node);
};
```

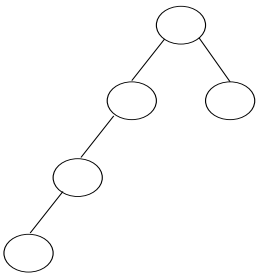
Abbildung 4: Schnittstelle eines ADT für Bäume

```
   unsigned int inhalt;
   BinBaum * links;
   BinBaum * rechts;
};
```

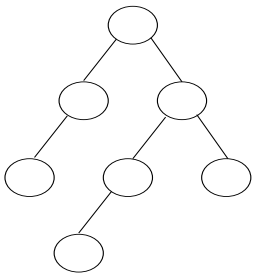
Eine Definition für die Höhe eines Baumes lautet: Die Höhe eines Baumes, der nur einen Knoten hat ist 1. Die Höhe eines beliebigen Baumes ist das Maximum der Höhen seiner Teilbäume plus eins. Ein Binärbaum heißt balanciert, wenn für jeden Knoten gilt, daß die Höhe des linken Teilbaums sich von der Höhe des rechten Teilbaums höchstens um den Wert eins unterscheidet. Hierzu einige Beispiele:



balanciert
Hoehe = 2



nicht balanciert
Hoehe = 4

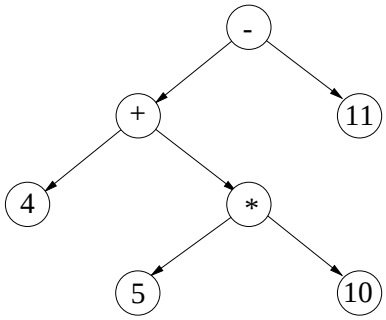


balanciert
Hoehe = 4

Schreiben Sie eine rekursive C++-Prozedur istBalanciert, die ermittelt, ob ein gegebener Binärbaum balanciert ist oder nicht.

25. Aufgabe :

Binäre Bäume eignen sich gut, um arithmetische Ausdrücke zu speichern. Die Blätter solcher Bäume enthalten die Zahlen (oder Variablen) des Ausdrucks, die inneren Knoten repräsentieren die Operatoren. Ein Beispiel: Der Ausdruck $(4 + 5 * 10) - 11$ ergibt den nachfolgenden Baum.



(a) Entwerfen Sie eine Datenstruktur Ausdruck, die einfache Ausdrücke baumartig speichert. Es soll nur mit Zahlen, nicht mit Variablen gerechnet werden. Als Operationen sind nur Addition, Subtraktion, Multiplikation und Division zugelassen (ganze oder reelle Zahlen).

(b) Implementieren Sie eine Funktion Auswertung, die eine Variable vom Typ Ausdruck als Eingabe erhält, den übergebenen Ausdruck auswertet und das Ergebnis zurückliefert.

(c) Geben Sie eine Grammatik an, mit der die oben angeführte Art von Ausdrücken beschrieben wird (auf Präzedenz achten — Punkt- vor Strichrechnung). Wie unterscheidet sich der Expression-Baum eines Ausdrucks von seinem Ableitungsbaum?

26. Aufgabe :

In der Vorlesung haben Sie verschiedene Realisierungsmöglichkeiten für Bäume mit beliebigem Auslaufgrad (N-äre Bäume) kennengelernt. Wir betrachten hier zwei davon: „Knoten mit festem Auslaufgrad und Überlaufwächter“ und „LeftmostChild-RightSibling“.

(a) Entwerfen Sie MODULA-2 Datentypen, die diesen Realisierungen entsprechen.

(b) Schreiben Sie für beide in a) entworfenen Datentypen eine Prozedur, die ein Kind unter einen Knoten einfügt.

(c) Schreiben Sie für beide in a) entworfenen Datentypen eine Prozedur, die die Höhe eines Baumes ermittelt.

Datenstruktur Graph

27. Aufgabe :

- Realisieren Sie ein Datenobjektmodul UnmarkierterGerichteterGraph mit Hilfe einer Adjazenzmatrixdarstellung. Dabei sei die maximale Anzahl von Knoten im Graphen konstant. Eine Adjazenzmatrix ist eine Matrix $AdjazenzM$ mit Booleschen Einträgen für die gilt:

$$AdjazenzM[i,j] = \begin{cases} \textit{TRUE} & \text{falls es eine Kante von Knoten i zu Knoten j im Graphen gibt} \\ \textit{FALSE} & \text{sonst} \end{cases}$$

- Wie müssen Sie Ihre Realisierung ändern / erweitern, wenn Kanten und / oder Knoten im Graphen markiert sein dürfen?

28. Aufgabe :

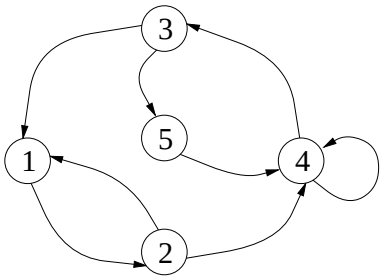
In Aufgabe 39 haben Sie ein Datenobjektmodul UnmarkierterGerichteterGraph erstellt, das einen Graphen mit Hilfe einer Adjazenzmatrix realisierte. Verwenden Sie dieses abstrakte Datenobjekt, um ein Datenobjektmodul UnmarkierterUngerichteterGraph zu entwerfen.

- Unterscheiden sich die Schnittstellen von UnmarkierterGerichteterGraph und UnmarkierterUngerichteterGraph? Begründen Sie.
- Schreiben Sie Prozeduren zum Einfügen und Löschen von Kanten in den ungerichteten Graphen, wie Sie im Rumpf des Datenobjektmoduls stehen könnten. Hat die Implementierung von UnmarkierterGerichteterGraph als Adjazenzmatrix Einfluß auf die Realisierung der Prozeduren?

29. Aufgabe :

In der Vorlesung haben Sie verschiedene Realisierungsmöglichkeiten für die knotenorientierte Speicherung (Adjazenz-Darstellung) von Graphen kennengelernt. Von den vorgestellten Möglichkeiten beinhaltet nur die charakteristische Adjazenzmatrix direkte Informationen über in Knoten einlaufende Kanten. Diese Information ist aber wichtig, wenn man z.B. das Löschen von Knoten effizient implementieren will (warum?).

(a) Entwerfen Sie drei Datenstrukturen, die die sequentiellen Adjazenzlisten, die verdichteten seq. Adjazenzlisten und die verketteten Adjazenzlisten so erweitern, daß zu jedem Knoten Informationen über seine einlaufenden Kanten gespeichert werden. Stellen sie die entworfenen Strukturen skizzenhaft dar (Zeichnungen analog zu denen im Skript). Als Beispiel für die Skizzen soll folgender Graph dienen.



(b) Geben Sie für eine der von Ihnen entworfenen Datenstrukturen eine entsprechende Typdefinition in C++ an.

(c) Wie groß ist der Aufwand zum herausfinden aller einlaufenden Kanten in einen Knoten bei den im Skript vorgestellten Realisierungsmöglichkeiten und wie groß ist er bei Ihren Datenstrukturen (Graph mit n Knoten und m Kanten)? Womit erkaufen Sie sich diesen Vorteil?

(d) Entwerfen Sie eine Datenstruktur, bei der nicht nur die Adjazenzlisten verkettet sind, sondern ebenfalls die Knoten des Graphen in einer verketteten Liste gespeichert werden. Wie sollten dabei sinnvollerweise die Ziel- bzw. Quellknoten in den Adjazenzlisten identifiziert werden? (Zeichnung und Typdefinition)

Module und Datenabstraktion in C++

30. Aufgabe :

Schreiben Sie in C++ die Schnittstelle eines Datenobjektmoduls, das eine Menge von Personendaten verwaltet. Das Modul soll die Operationen Einfuegen, Loeschen und Suchen anbieten.

Im Rumpf dieses Moduls sollen die Personendaten in einer doppelt verketteten Liste nach den Nachnamen sortiert gespeichert werden. Um einen effizienteren Zugriff auf die Elemente der Liste zu haben, wird statt

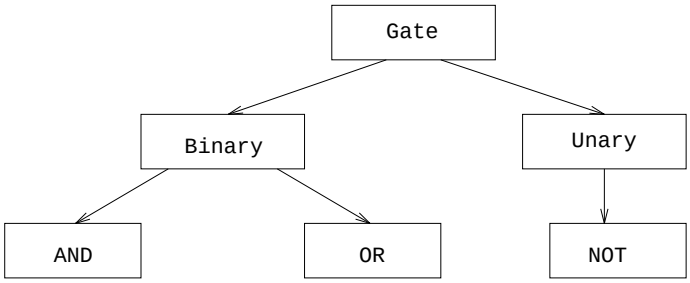
eines einzelnen Ankerzeigers ein Feld von 26 Ankern vereinbart, die jeweils auf die erste Person in der Liste, deren Nachname mit einem bestimmten Buchstaben des Alphabetes beginnt, zeigen sollen. Implementieren Sie folgende Teile des Rumpfes: die Deklarationen der benötigten Variablen und die Funktion zum Einfuegen von Personendaten.

31. Aufgabe :

Definieren Sie in C++ eine Klasse BinBaum, die einen abstrakten Datentyp „Binärer Baum“ verkapselt. Die Klasse sollte Methoden zum Einfügen und Löschen von Knoten sowie zum Navigieren durch den Baum besitzen (vgl. Aufgabe 35).

32. Aufgabe :

Im folgenden sollen einfache logische Schaltungen simuliert werden. Dazu werden einige Klassen in C++ definiert, die diese Bauteile nachbilden. Die Basisklasse ist die Klasse Gate. Weitere Schaltelemente sollen gemäß der folgenden Klassenhierarchie ebenfalls vereinbart werden:



Wir unterscheiden also Gatter mit zwei Eingängen und Gatter mit einem Eingang. Alle Gatter haben allerdings nur einen Ausgang. Um aus solchen Objekten kleine Schaltnetze aufbauen zu können, soll jede Klasse die Methoden void setInput(int n, logic value) (setzen des Eingangswertes für Eingang Nr. n) und logic getOutput(void) (auslesen des aktuellen Ausgabewertes) besitzen. Der Typ logic sei dabei ein Aufzählungstyp mit den Werten 0 und 1.

- Implementieren Sie die angegebenen Klassen. Dabei sollen die Methoden setInput und getOutput virtuell definiert werden.
- Definieren Sie eine Klasse Halfadder, deren Objekte sich aus den für einen Halbaddierer benötigten Gatter-Objekten zusammensetzen. Ein Halfadder soll ebenfalls Methoden zum setzen der Eingabewerte und zum Auslesen der Ausgabewerte (allerdings hier für zwei Ausgänge) besitzen. Die Ausgabewerte sollen mit Hilfe der Gatter-Objekte berechnet werden.

33. Aufgabe :

Implementieren Sie in C++ eine Klasse ListenElement und eine Klasse Liste. Objekte vom Typ Liste sollen Objekte vom Typ ListenElement (und davon abgeleitete) in einer doppelt verketteten Liste speichern können. Schreiben Sie für Liste einen Konstruktor, der Objekte als leere Listen initialisiert und einen Destruktor, der alle in der Liste gespeicherten Objekte ebenfalls zerstört.

34. Aufgabe $\star$:

In C++ können Klassen auch von mehr als einer Basisklasse abgeleitet werden (Mehrfachvererbung). Die Definition einer Klasse C, die von zwei Basisklassen A und B erbt, sieht so aus:

class C: A, B {...}

Betrachten Sie nun die folgenden Fortbewegungsmittel: Amphibienfahrzeug, Cabrio, Fahrzeug, Landfahrzeug, LKW, Motorboot, PKW, Segelboot, Wasserfahrzeug, Yacht.

- Stellen Sie für diese Begriffe eine sinnvolle Klassenhierarchie auf, in der auch Mehrfachvererbung vorkommt.
- Jede Klasse soll eine Methode besitzen, mit der man den jeweiligen Fahrzeugtyp erfragen kann. Definieren Sie diese Methode als virtuelle Methode in der Klasse Fahrzeug und redefinieren Sie sie in einer der Unterklassen.