Lösung zur
Vordiplom Informatik für E-Technik

SS 2000

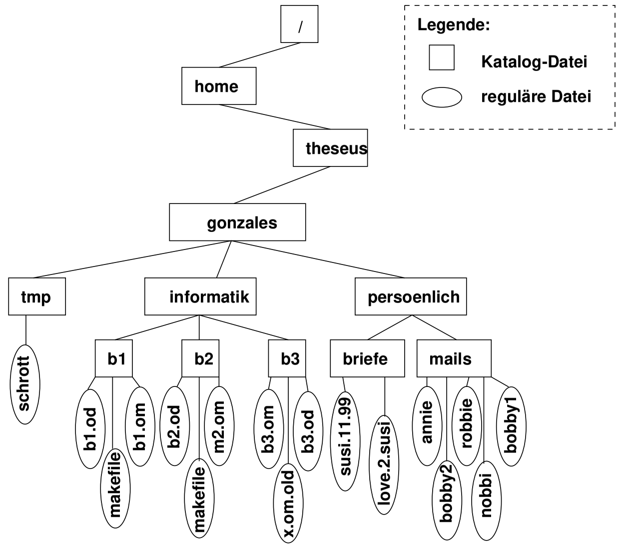
22. September 2000

Aufgabe		Punkte	Bewertung
1	a)	4	
	b)	6	
2	a)	4	
	b)	3	
	c)	3	
3		10	
4	a)	2	
	b)	3	
	c)	3	
	d)	2	
5	a)	4	
	b)	6	
6	a)	4	
	b)	6	
7	a)	2	
	b)	5	
	c)	3	
Summe		**70**	

Aufgabe 1 (18 Punkte)

Dateisystem, Pipes & Filters, einfache Kommandos

Der Benutzer `gonzales` auf einem UNIX-Rechner ist nach dem Anmelden in seinem Heimatkatalog `/home/theseus/gonzales`. Dort findet sich die im folgenden Bild dargestellte Struktur wieder:



(*a*) 2 Punkte

Geben Sie an, wie `gonzales` ohne den Arbeitskatalog zu wechseln die Datei `schrott` löschen kann:

```
theseus$ rm tmp/schrott
```

(*b*) 2 Punkte

`gonzales` möchte den Katalog `mails` zum Arbeitskatalog machen:

```
theseus$ cd persoenlich/mails
```

(*c*) 2 Punkte

`mails` ist nun der Arbeitskatalog und `gonzales` möchte alle Dateien, in den die Buchstabenfolge `obb` vorkommt, in seinen Katalog `tmp` kopieren. Da er schreibfaul ist, möchte er die 4 Dateinamen, die in Frage kommen, nicht einzeln eintippen. Geben Sie ihm eine möglichst kompakte Form:

```
theseus$ cp *obb* ../../tmp
```

(*d*) 2 Punkte

Bei der Ausführung des Kommandos in Teilaufgabe c) erhält er die Diagnosemeldung `cp: nobbi: permission denied`. Nennen Sie eine mögliche Ursache hierfür!

Das Leserecht der Datei `nobbi` ist nicht gesetzt!

(*e*) 6 Punkte

`gonzales` ist inzwischen in den Katalog `b3` gewechselt. Das hier enthaltene Oberonprogramm soll nun möglichst effizient übersetzt werden (also über ein `makefile`). Nach der Übersetzung sollen alle vom Compiler erzeugten Hilfsdateien gelöscht werden, das ausführbare Programm soll allerdings erhalten bleiben für Teilaufgabe f). Geben Sie alle notwendigen Schritte an (die Anzahl der folgenden Eingabezeilen sagt nichts über die Anzahl der tatsächlich benötigten Kommandos aus):

```
theseus$ mmo > makefile

theseus$ make

theseus$ make clean

theseus$ ________________________________

theseus$ ________________________________
```

(*f*) 4 Punkte

Das Programm aus Teilaufgabe e) ist ein Programm zum Verschlüsseln von Texten und es ist als Filter konstruiert. `gonzales` will es mit der relativ großen Datei `susi.11.99` (mehr als eine Bildschirmseite) "füttern" und das Ergebnis in aller Ruhe ansehen (Hinweis: Kommando `more`). Geben Sie die notwendige Kommandozeile an:

```
theseus$ < ../../persoenlich/briefe/susi.11.99 b3 | more
```

oder

```
theseus$ cat ../../persoenlich/briefe/susi.11.99 | b3 | more
```

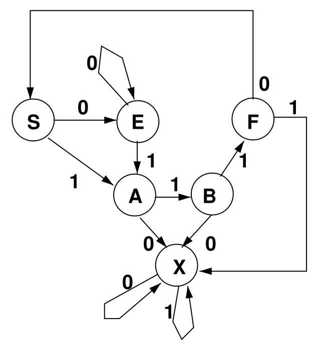

Aufgabe 2 (12 Punkte)

Formale Sprachen, endliche Automaten und reguläre Ausdrücke

Über dem Vokabular V = {0,1} sei eine formale Sprache definiert als die Menge aller nicht-leeren Symbolfolgen, bei denen die Teilfolgen, die nur das Symbol '1' beinhalten, die Länge 3 haben.
Beispiele für korrekte Folgen: 0, 1110000, 11100111, 00000
Beispiele für nicht korrekte Folgen: 110000, 111001111000

(*a*) 4 Punkte

Geben Sie in grafischer Form einen endlichen Automaten an, der diese Sprache definiert. Dabei ist durch einen Fehlerzustand auch die Behandlung nicht korrekter Folgen zu berücksichtigen!



(*b*) 4 Punkte

Beschreiben Sie diese Sprache durch eine reguläre Grammatik!

S	**–>**	**0E**	**S**	**–>**	**1A**	**E**	**–>**	**0E**
E	**–>**	**1A**	**A**	**–>**	**1B**	**B**	**–>**	**1F**
F	**–>**	**0S**	**F**	**–>**		**E**	**–>**	

(*c*) 4 Punkte

Geben Sie einen regulären Ausdruck mit den in der Vorlesung verwendeten Metasymbolen an, der diese Sprache definiert!

(00* | 111)(00*111)*0*

Aufgabe 3 (10 Punkte)

Programmiertechniken

(*a*) 2 Punkte

Programme, die mit reellen Zahlen (Datentyp `REAL` in Oberon) arbeiten, können, obwohl mathematisch korrekt, fehlerhafte Ergebnisse liefern.

- ☐ Worauf ist bei der Addition von sehr vielen Daten in sehr unterschiedlicher Größenordnung zu achten?
  Antwort:

  Zahlen sortieren und von den kleinen Zahl her aufaddieren

- ☐ x und y seien zwei reelle Variable, deren Werte wie auch immer berechnet wurden. Wie ist die Bedingung *Wenn x und y gleich sind, dann ...* sinnvollerweise zu formulieren?
  Antwort:

  Es ist eine Genauigkeitsschranke *epsilon* zu definieren und die Abfrage als `IF ABS(x-y) < epsilon THEN` **zu formulieren.**

(*b*) 4 Punkte

- ☐ Was versteht man unter *Information Hiding*?
  Antwort:

  Verbergen von implementierungsspezifischen Details und Bereitsstellen nur der Informationen, die zur Benutzung gebraucht werden.

- ☐ Nennen Sie zwei Ziele, die mit der Technik des *Information Hiding* angestrebt werden!
  Antwort:

  Flexibilität in der Implementierung, Erhöhung der Verständlichkeit, der Prüfbarkeit, der ...

(*c*) 4 Punkte

Skizzieren Sie am Beispiel eines *Stack*, was man unter einem *Abstrakten Datentyp* versteht und wie dieses Konzept in Oberon umgesetzt werden kann!
Antwort:

Im Definitions-Modul wird nur der Name des Datentyps (hier z.B. `Stack`**) sowie die Prozedurköpfe der Operationen auf dem Datentyp bekanntgegeben. Als Basistyp für die Elemente, die auf dem Stack verwaltet werden, wird** `Objects.Object` **verwendet.**

Aufgabe 4 (8 Punkte)

Prozedur-Parameter

(*a*) 4 Punkte

Gegeben sei folgendes Oberon-Programm:

```
MODULE Obscure;
    IMPORT Write;
    VAR x,y: INTEGER;

    PROCEDURE Magic(VAR x: INTEGER; y: INTEGER);
    BEGIN
       x := x-y; y := y-x
    END Magic;

BEGIN
    x := 24; y:= 10;
    WHILE x # y DO
      IF x >= y + 1 THEN
         Magic(x,y)
      ELSE
         Magic(y,x)
      END;
      Write.String("x = "); Write.Int(x,0);
      Write.String(" | ");
      Write.String("y = "); Write.Int(y,0);
      Write.Ln;
   END
END Obscure.
```

Geben Sie exakt an, was dieses Programm als Ausgabe liefert!

Lösung:

```
x =  14 | y =  10
x =   4 | y =  10
x =   4 | y =   6
x =   4 | y =   2
x =   2 | y =   2
```

(*b*) 4 Punkte

Schreiben Sie eine Oberon-Prozedur `Count`, die als Parameter einen beliebigen String (entsprechend den Oberon- / Unix-Konventionen) und ein Zeichen bekommt und als Rückgabewert die Häufigkeit dieses Zeichens in diesem String liefert!

Lösung:

```
PROCEDURE CharCount(s: ARRAY OF CHAR; c: CHAR):INTEGER;
   VAR i,z: INTEGER;
BEGIN
   i := 0; z := 0;
   WHILE (i < SHORT(LEN(s))) & (s[i] # ASCII.nul) DO
      IF s[i] = c THEN INC(z) END;
      INC(i)
   END;
   RETURN z
END CharCount;
```


Aufgabe 5

Zeichenkonstante & Arrays & Argumentverarbeitung

Es ist ein vollständiges Oberon-Programm zu schreiben, das als Optionen (*Flags*) alle Kleinbuchstaben (keine Umlaute) erlaubt. Andere Zeichen sollen mit dem Hinweis "Illegal Option: <zeichen>" ausgegeben werden und nicht zum Programmabbruch führen. Das Programm soll in einem Array die Information, ob eine erlaubte Option gesetzt ist oder nicht, aufsammeln und nach vollständiger Bearbeitung der Flags in alphabetischer Ordnung ausgeben, welche Flags gesetzt und welche nicht gesetzt sind.

Hinweis: Überlegen Sie zunächst, wie die Menge der Kleinbuchstaben auf den Indexbereich eines Arrays abgebildet werden kann!

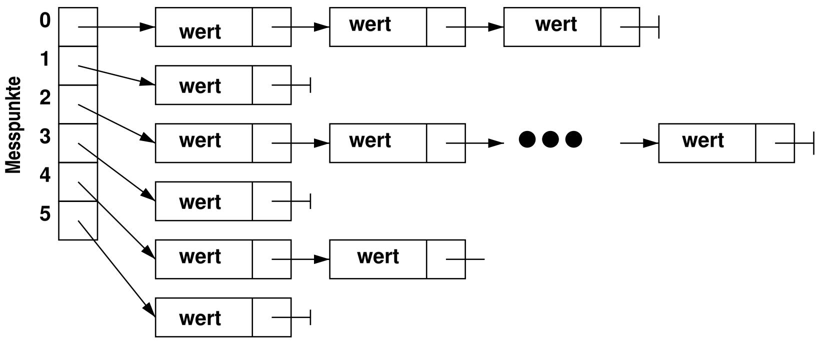
Lösung:

```
MODULE MyArgs;
  IMPORT Write, UA := UnixArguments, Conclusions;
  VAR
      flags: ARRAY 26 OF BOOLEAN;
      ch: CHAR;
      i: INTEGER;

BEGIN
     i := 0;
     WHILE i < 26 DO
       flags[i] := FALSE;
       INC(i);
     END;

     WHILE UA.GetFlag(ch) DO
       IF ("a" <= ch) & (ch <= "z") THEN
          flags[ORD(ch)-ORD("a")] := TRUE;
       ELSE
          Write.String("Illegal Option: ");
          Write.Char(ch); Write.Ln;
       END;
     END;

     i := 0;
     ch := "a";
     WHILE i < 26 DO
       Write.Char(ch);
       IF flags[i] THEN
          Write.String(" ist gesetzt!"); Write.Ln
       ELSE
          Write.String(" ist nicht gesetzt!"); Write.Ln
       END;
       INC(i); ch := CHR(ORD(ch)+1);
     END;
END MyArgs.
```

Aufgabe 6 (4 Punkte)

Dynamische Datenstrukturen

In einem technischen System sind eine bestimmte Anzahl von Messpunkten definiert, an denen im Testbetrieb jeweils Messungen (in unterschiedlicher, und vorab nicht bekannter Zahl) durchgeführt werden. Um dieses Messdaten speichern zu können, hat sich "jemand" folgende Datenstruktur überlegt:



Die Messwerte (in der Zeichnung "wert") sind vom Typ `REAL`, Die Anzahl der Messpunkte sei durch die Konstante `Anzahl` definiert.

Geben Sie für diese Datenstruktur alle notwendigen Typvereinbarungen an, so dass eine Variable `messReihe` für eine konkrete Messung vereinbart werden kann!

Lösung:

```
CONST Anzahl = 6;
TYPE
      Element = POINTER TO ElementRec;
      ElementRec = RECORD
                     wert: REAL;
                     next: Element;
                   END;
      MessReihe = ARRAY Anzahl OF Element;
```


Aufgabe 7 (8 Punkte)

Streams

Sie erhalten eine Datei `messwerte` -- zu lesen durch Eingabeumlenkung -- mit folgendem Aufbau:

- Jede Zeile (Zeilentrenner ist *newline*) enthält mindestens eine, ansonsten aber beliebig viel INTEGER-Zahlen, die durch *blanks* oder *tabs* getrennt sind.
- Die maximale Länge einer Zeile in Bytes gemessen sei 1024 (inkl. *newline*).
- Die Zahlen der i-ten Zeile sind Messwerte am Messpunkt i.

Ergänzen Sie untenstehendes OberonProgramm so, dass als Ausgabe die Folge <messpunkt anzahlMesswerte> entsteht! Bei folgender Eingabedatei

12 14 15 223 44
12
14 15
100
17 18

soll das Programm folgende Ausgabe liefern:

```
1  5
2  1
3  2
4  1
5  2
```

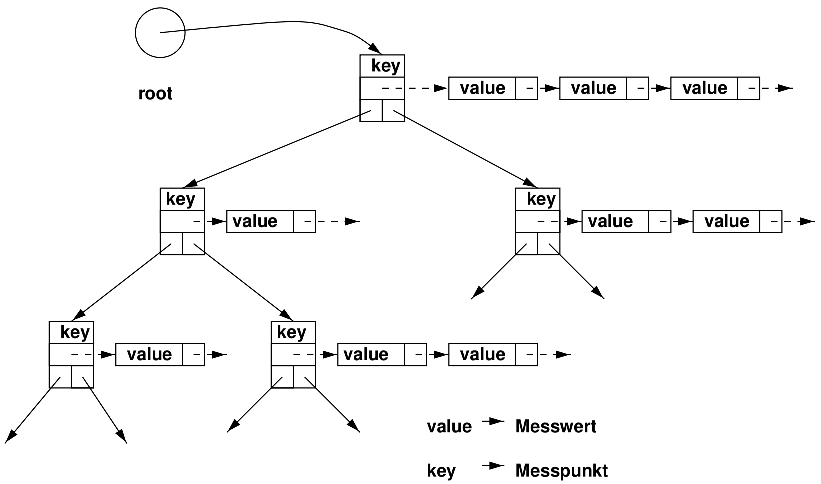
Lßoung:

```
MODULE Count;
   IMPORT Streams, Strings, Read, Write;
   TYPE
      Line = ARRAY 1024 OF CHAR;
   VAR
      line: Line; value, valueCount, lineNo: INTEGER;
      str: Streams.Stream;
BEGIN
      lineNo := 1;
      Read.Line(line);
      WHILE Streams.stdin.count > 0 DO
         valueCount := 0;
         Strings.Open(str, line);
         Read.IntS(str, value);
         WHILE str.count > 0 DO
           INC(valueCount);
           Read.IntS(str, value);
         END;
         IF Streams.Close(str) THEN END;
         Write.Int(lineNo,4); Write.Int(valueCount, 4); Write.Ln;
         INC(lineNo);
         Read.Line(line);
      END;

END Count.
```

Aufgabe 8 (20 Punkte)

Dynamische Datenstrukturen und Rekursion

Die Verwaltung von Messwerten an verschiedenen Messpunkten könnte auch mit einem sortierten Binärbaum erfolgen:



Dazu seien folgende Typvereinbarungen vorgegeben:

```
TYPE
  Tree     = POINTER TO TreeNode;
  List     = POINTER TO ListNode;
  TreeNode = RECORD
               key: INTEGER;
               values: List;
               left, right: Tree;
             END;
  ListNode = RECORD
               value: INTEGER;
               next: List;
             END;
```

Für die folgenden Aufgaben können Sie davon ausgehen, dass der zugrundeliegende Baum bzgl. der Komponente `key` nach **inorder sortiert** ist.

(*a*) 10 Punkte

Schreiben Sie eine **rekursive** Oberon-Prozedur, die als Parameter den Wert eines Messpunktes (Komponente `key` im Typ `TreeNode`) und einen zugehörigen Messwert (Komponente `value` im Typ `ListNode`) sowie den Zeiger auf die Wurzel eines konkreten Baumes erhält und den Messwert in die Liste, die durch den Messpunkt definiert ist, einfügt (wo eingefügt wird, ist nicht vorgegeben)!

Lösung:

```
PROCEDURE AddValue(key, value: INTEGER; root: Tree);
    VAR val: List;
BEGIN
   IF root = NIL THEN RETURN
   ELSIF root.key = key THEN
      NEW(val);
      val.value := value;
      val.next := root.values;
      root.values := val;
   ELSIF root.key < key THEN
       AddValue(key,value,root.right)
   ELSE
       AddValue(key,value,root.left)
   END
END AddValue;
```

(*b*) 10 Punkte

Schreiben Sie eine **rekursive** Oberon-**Funktionsprozedur**, die die Anzahl aller Messwerte liefert!

Lösung:

```
PROCEDURE CountValues(root: Tree):INTEGER;
  VAR z: INTEGER;
      list: List;
BEGIN
  IF root = NIL THEN RETURN 0
  ELSE
     z :=0;
     list := root.values;
     WHILE list # NIL DO
        INC(z); list := list.next;
     END;
     RETURN z + CountValues(root.left) + CountValues(root.right)
  END;
END CountValues;
```