Professor Dr. F. Schweiggert   Abteilung Angewandte Informationsverarbeitung   25. Januar 2001
Johannes Mayer

Allg. Informatik für WiWi (WS 2000/2001)

Abgabetermin: 25. Januar 2001

Beispiellösung

1 Automat (10 Punkte)

In Aufgabe 3b auf Blatt 4 mussten Sie zu einer gegebenen regulären Sprache einen endlichen Automaten konstruieren. Nun sollen Sie diesen Automaten durch ein Oberon-Programm implementieren. Dieses Programm soll Zeile für Zeile von der Standardeingabe (*Beispiel*) einlesen (bis „end of file" erreicht ist) und danach soll jeweils überprüft werden, ob der Satz (ohne Zeilentrenner), d. h. eine Zeile, zur Sprache gehört und entsprechend „ja" oder „nein" ausgegeben werden.
(Hinweise: Denken Sie bei der Implementierung auch an den impliziten Fehlerzustand des Automaten, d. h. alle nicht spezifizierten Übergänge führen in den Fehlerzustand. Außerdem treten hier nicht nur die Zeichen „0" und „1", sondern beliebige ASCII-Zeichen auf. Sie können Ihren (korrekten!) Automaten oder den Automaten aus der Beispiellösung zu Aufgabe 3b von Blatt 4 als Grundlage verwenden. Sie dürfen davon ausgehen, dass in jeder Zeile maximal 100 Zeichen eingegeben werden.)

Lösung:

```
DEFINITION Automat;
END Automat.
```

```
MODULE Automat;

  IMPORT Read, Streams, Write;

  CONST
    MaxZeilenLaenge = 100;

    (* Zustaende des Automaten aus der Beispielloesung
     * von Aufgabe 3b auf Blatt 4 *)
```

```
    S = 0;
    A = 1;
    B = 2;
    C = 3;
    D = 4;
    E = 5;
    Fehler = 6;    (* impliziter Fehlerzustand des Automaten *)

  TYPE
    (* String enthaelt am Ende ein Null-Zeichen (0X)
     * => Array der Laenge MaxZeilenLaenge + 1 *)
    String = ARRAY MaxZeilenLaenge + 1 OF CHAR;

  VAR
    zeile      : String;
    zustand, i : INTEGER;

BEGIN
  Read.Line(zeile);
  WHILE ~ Streams.stdin.eof DO
    zustand := S;
    i := 0;

    WHILE zeile[i] # 0X DO

      (* eigentliche Implementierung des Automaten *)
      CASE zustand OF
      | S : CASE zeile[i] OF
            | "1" : zustand := A;
            ELSE
              zustand := Fehler;
            END;
      | A : CASE zeile[i] OF
            | "1" : zustand := B;
            ELSE
              zustand := Fehler;
            END;
      | B : CASE zeile[i] OF
            | "0" : zustand := C;
            ELSE
              zustand := Fehler;
            END;
      | C : CASE zeile[i] OF
            | "0" : zustand := D;
            | "1" : zustand := E;
            ELSE
```

```
          zustand := Fehler;
        END;
    | D : CASE zeile[i] OF
        | "0" : zustand := C;
        ELSE
          zustand := Fehler;
        END;
    | E : zustand := Fehler;
    ELSE
      (* im Zustand Fehler bleiben - egal welches Zeichen kommt *)
    END;

    INC(i);
  END;

  CASE zustand OF
  | E : Write.String("ja"); Write.Ln;
  ELSE
    Write.String("nein"); Write.Ln;
  END;

  Read.Line(zeile);
 END;
END Automat.
```

2 Datenstrukuren - Arrays vs. Records (20 Punkte)

Nun haben Sie die Aufgabe Studentendaten des Studentenwerks zu verwalten. Für jeden Studenten bekommen Sie seinen Namen (max. 80 Zeichen), seine Matrikelnummer (positive ganze Zahl kleiner oder gleich 9999999) und sein Geburtsdatum, das aus Tag, Monat und Jahr besteht.

(a) Definieren sie zuerst Oberon-Datenstrukturen, um einen Studenten mit den angegebenen Daten zu repräsentieren. Verwenden Sie hierbei Arrays und Records. Die Datenstruktur soll es ermöglichen, direkt auf Tag, Monat und Jahr des Geburtsdatums zuzugreifen.
(5 Punkte)

(b) Jetzt bekommen Sie eine kleine *Studentendatenbank*. In jeder Zeile befinden sich die Daten eines Studenten. Name, Matrikel-Nummer und Geburtsdatum sind jeweils durch einen Doppelpunkt getrennt. Das Datum ist im Format `[d]d.[m]m.[yy]yy` angegeben (siehe Blatt 6). Es kommen keine unnötigen Whitespaces (d. h. Leerzeichen, Tabulator oder Zeilentrenner) vor.
Beispiel (für eine Zeile): `Thomas Huber:123456:13.2.1979`
Schreiben Sie ein Oberon-Programm, das von der Standardeingabe die beschriebene Datenbank (mit maximal 100 Datensätzen) in ein Array der Datenstruktur aus (a) einliest (nach dem letzten Datensatz kommt das „end of file") und dann den ältesten Studenten sucht und ausgibt.

(Hinweise: Sie können davon ausgehen, dass die Eingabe genau die beschriebene Form besitzt. Ist die Jahreszahl nur zweistellig angegeben, so soll 19 als Jahrhundert angenommen werden. Sie können davon ausgehen, dass es mindestens einen Datensatz gibt. Bei mehreren gleich alten ältesten Studenten geben Sie einen beliebigen aus.)
(15 Punkte)

Lösung:

(a)
```
CONST MaxStringLaenge = 80;

TYPE String = ARRAY MaxStringLaenge + 1 OF CHAR;
     Datum = RECORD
               tag,
               monat,
               jahr  : INTEGER;
             END;
     Student = RECORD
                 name           : String;
                 matrikelNummer : INTEGER;
                 geburtsDatum   : Datum;
               END;
```

(b)
```
DEFINITION Studenten;
END Studenten.
```

```
MODULE Studenten;

  IMPORT Read, Streams, Write;

  CONST MaxStringLaenge = 80;
        MaxDatensaetze = 100;
        MaxZeile = 120;

  TYPE String = ARRAY MaxStringLaenge + 1 OF CHAR;
       Datum = RECORD
                 tag,
                 monat,
                 jahr  : INTEGER;
               END;
       Student = RECORD
                   name           : String;
                   matrikelNummer : INTEGER;
                   geburtsDatum   : Datum;
                 END;
```

```
      Datenbank = ARRAY MaxDatensaetze OF Student;

      Zeile = ARRAY MaxZeile + 1 OF CHAR;

  VAR db : Datenbank;
      i, j, aeltester : INTEGER;
      zeile : Zeile;
BEGIN
  (*** DATENBANK einlesen in db ***)

  i := -1; (* Index des aktuellen Datensatzes *)
  Read.Line(zeile);
  WHILE ~ Streams.stdin.eof DO
    INC(i);

    j := 0; (* Index des aktuellen Zeichens in der Zeile *)

    (* Feld NAME lesen *)

    WHILE zeile[j] # ":" DO
      db[i].name[j] := zeile[j];
      INC(j);
    END;
    db[i].name[j] := 0X; (* Name mit Null-Zeichen abschliessen *)

    INC(j); (* ":" ueberlesen *)

    (* Feld MATRIKELNUMMER lesen *)

    db[i].matrikelNummer := 0;
    WHILE zeile[j] # ":" DO
      db[i].matrikelNummer := db[i].matrikelNummer * 10 +
                              (ORD(zeile[j]) - ORD("0"));
      INC(j);
    END;

    INC(j); (* ":" ueberlesen *)

    (* Feld GEBURTSDATUM.TAG lesen *)

    db[i].geburtsDatum.tag := 0;
    WHILE zeile[j] # "." DO
      db[i].geburtsDatum.tag := db[i].geburtsDatum.tag * 10 +
                                (ORD(zeile[j]) - ORD("0"));
      INC(j);
    END;
```

```
  INC(j); (* "." ueberlesen *)

  (* Feld GEBURTSDATUM.MONAT lesen *)

  db[i].geburtsDatum.monat := 0;
  WHILE zeile[j] # "." DO
    db[i].geburtsDatum.monat := db[i].geburtsDatum.monat * 10 +
                                (ORD(zeile[j]) - ORD("0"));
    INC(j);
  END;

  INC(j); (* "." ueberlesen *)

  (* Feld GEBURTSDATUM.JAHR lesen *)

  db[i].geburtsDatum.jahr := 0;
  WHILE zeile[j] # 0X DO
    db[i].geburtsDatum.jahr := db[i].geburtsDatum.jahr * 10 +
                               (ORD(zeile[j]) - ORD("0"));
    INC(j);
  END;

  (* ggf. Jahrhundert hinzufuegen *)
  IF db[i].geburtsDatum.jahr < 100 THEN
    db[i].geburtsDatum.jahr := db[i].geburtsDatum.jahr + 1900;
  END;

  Read.Line(zeile);
END;

(* jetzt steht in i der Index des letzten Datensatzes *)

(*** aeltesten Studenten in db suchen ***)

aeltester := 0; (* Index des bisher aeltesten *)
j := 1;

WHILE j <= i DO
  (* teste, ob Student Nr. j aelter als Student Nr. aeltester *)
  IF (db[j].geburtsDatum.jahr < db[aeltester].geburtsDatum.jahr)
     OR ((db[j].geburtsDatum.jahr = db[aeltester].geburtsDatum.jahr) &
         (db[j].geburtsDatum.monat < db[aeltester].geburtsDatum.monat))
     OR ((db[j].geburtsDatum.jahr = db[aeltester].geburtsDatum.jahr) &
         (db[j].geburtsDatum.monat = db[aeltester].geburtsDatum.monat) &
         (db[j].geburtsDatum.tag < db[aeltester].geburtsDatum.tag))
```

```
    THEN
      aeltester := j;
    END;

    INC(j);
  END;

  (*** aeltesten Studenten ausgeben ***)
  Write.String("aeltester Student:"); Write.Ln;
  Write.String("=================="); Write.Ln;
  Write.Ln;
  Write.String(db[aeltester].name); Write.Ln;
  Write.Int(db[aeltester].matrikelNummer, 0); Write.Ln;
  Write.Int(db[aeltester].geburtsDatum.tag, 0); Write.Char(".");
  Write.Int(db[aeltester].geburtsDatum.monat, 0); Write.Char(".");
  Write.Int(db[aeltester].geburtsDatum.jahr, 0); Write.Ln;
END Studenten.
```