# Aufgabe 1

In einer Firma wird die Arbeitszeit über ein Stechkartensystem erfaßt. Bei jeder Passage der Eingangstür werden die folgenden Daten erfaßt:

| | |
|---|---|
| Namen der Person | 20 Zeichen |
| Datum und Uhrzeit | JJJJMMTTssmm, Zeitform als Zeichenkette |
| | JJJJ = 4 Ziffern für Jahr |
| | MM = 2 Ziffern für Monat |
| | TT = 2 Ziffern für Tag |
| | ss = 2 Ziffern für Stunde |
| | mm = 2 Ziffern für Minute |
| Richtung der Passage | c, ein Zeichen |

Um die Aufgabe zu vereinfachen wollen wir annehmen, daß

- nur ein Eintritt und ein Austritt pro Tag
- keine Arbeit über Mitternacht

möglich sind. Die Datensätze stehen in dem **Textfile Passagen**, etwa in der Form:

```
Lehmann              200001030730E
Sch    e             200001030732E
Sch    e             200001031527A
Lehmann              200001031530A
Lehmann              200001040730E
Sch    e             200001040732E
Sch    e             200001041527A
Lehmann              200001041530A
```

Schreiben Sie ein PASCAL-Programm, das eine einfach verkettete, alphabetisch sortierte Liste der Namen aufbaut. An jeden Knoten dieser Liste ist eine einfach verkettete Liste der Türpassagen anzuhängen, wobei auch ein Zeiger auf den letzten Knoten dieser Liste vorzusehen ist, um nicht bei jedem Anfügen die gesamte Liste durchlaufen zu müssen.

Es entsteht etwa die folgende Struktur:

```
Knoten der Personenliste:

+-----------------+---------------+
| Name der Person | erste Passage |------>
+-----------------+---------------+
| naechste Person | letzte Pasage |------>
+-----------------+---------------+
         |
         |
         V


Knoten der Passagenliste:

+-------------------------------+
| Datum und Uhrzeit der Passage |
+------------+------------------+
| Richtung   | naechste Passage |------>
+------------+------------------+
```

```
Struktur der Listen
(Die Personenliste vertikal am linken Rand hier als zyklische Liste
realisiert, die Passagenlisten horizontal nach rechts angehängt):


  +-------+
  |       |
  |       V
  |     +---+---+
  |     |   |   |---|
  |     +---+---+
  |     |   |   |---|
  |     +---+---+
  |       |          +-------+    +-------+      +-------+
  |       V          | ..... |    | ..... |      | ..... |
  |     +---+---+    +---+---+    +---+---+      +---+---+
  |     |...|   |--->|...|   |--->|...|   |----->|...|   |---|
  |     +---+---+    +---+---+    +---+---+  +-->+---+---+
  |     |   |   |----------------------------+
  |     +---+---+
  |       |            +-------+
  |       V            | ..... |
  |     +---+---+      +---+---+
  |     |...|   |----->|...|   |---|
  |     +---+---+  +-->+---+---+
  |     |   |   |--+
  |     +---+---+
  |       |          +-------+      +-------+
  |       V          | ..... |      | ..... |
  |     +---+---+    +---+---+      +---+---+
  |     |...|   |--->|...|   |----->|...|   |---|
  |     +---+---+    +---+---+  +-->+---+---+
  |     |   |   |---------------+
  |     +---+---+
  |       |
  +-------+
```

## Anforderungen :

- **Au ahlungs atentyp** für die Richtung der Passage
- **Funktion** zur Erzeugung eines Knotens für die Personenliste mit Eintragung der Daten für den Namen und die Zeiger, wobei die Schreibweise des Namens und der Zeiger auf den Nachfolger als Eingangsparameter übergeben werden und der Zeiger auf den erzeugten Knoten zurückgegeben wird (s. Aufgabe 2).
- **Funktion** zur Erzeugung eines Knotens für die Passagenliste mit Einlesen der Daten und Eintragung in den erzeugten Knoten, wobei der Zeiger auf den erzeugten Knoten zurückgegeben wird (s. Aufgabe 3).
- **Funktion** zum Suchen eines Namens in der Personenliste und Rückgabe des Zeigers auf den Vorgänger des vorhandenen Namens oder der Einfügestelle, wobei der Zeiger auf die Personenliste und die Schreibweise des Namens als Eingangsparameter übergeben werden (s. Aufgabe 4).
- **Pro e ur** zum Einfügen eines Namens in die Personenliste, wobei der Zeiger auf den Vorgänger und die Schreibweise des Namens als Eingangsparameter übergeben werden (s. Aufgabe 5).
- **Pro e ur** zum Anfügen einer Passage an die Passagenliste der referierten Person, wobei der Zeiger auf die Person als Eingangsparameter übergeben wird (s. Aufgabe 6).

- **Funktion**, die die Arbeitszeit der Mitarbeiter berechnet und ausgibt, sowie die Daten auf Zulässigkeit überprüft. Dabei wird der Zeiger auf die Personenliste als Eingangsparameter übergeben. Im Fehlerfall wird **alse** zurückgeben, sonst **true** (s. Aufgabe 7).

- **Pro e ur**, die zu einem gegebenen Zeitpunkt feststellt, welche Person(en) zu diesem Zeitpunkt

  - im Betrieb anwesend waren
  - gerade zum Dienst erschienen sind
  - den Betrieb gerade verlassen haben

  oder ob gerade niemand im Betrieb anwesend war. Dabei werden der Zeiger auf die Personenliste und die Zeit als Eingangsparameter übergeben (s. Aufgabe 8).

- **Hauptprogramm**, das unter Verwendung dieser Prozeduren und Funktionen aus den Eingabedaten die Datenstruktur aufbaut, bis der Eingabestrom des Files **Passagen** erschöpft ist, anschließend die Arbeitszeit des Personals durch Anwendung der Funktion aus Aufgabe 7 berechnet und im Erfolgsfall zu jedem über die Standardeingabe **input** eingegebenen Zeitpunkt die Prozedur aus Aufgabe 8 aufruft.

Schreiben Sie den Programmkopf, den Vereinbarungsteil und das Hauptprogramm. Bei der Eingabe von

```
200001010000
200001030730
200001030732
200001031200
200001031527
200001031530
200001040730
200001040732
200001041200
200001041527
200001041530
200001051530
```

über die Standardeingabe **input** sollte die Ausgabe etwa folgendermaßen aussehen:

```
Lehmann
    03.01.2000 von 07.30 Uhr bis 15.30 Uhr: Arbeitszeit 8 Stunden 0 Minuten
    04.01.2000 von 07.30 Uhr bis 15.30 Uhr: Arbeitszeit 8 Stunden 0 Minuten
    Gesamtarbeitszeit: 16 Stunden 0 Minuten

Schulze
    03.01.2000 von 07.32 Uhr bis 15.27 Uhr: Arbeitszeit 7 Stunden 55 Minuten
    04.01.2000 von 07.32 Uhr bis 15.27 Uhr: Arbeitszeit 7 Stunden 55 Minuten
    Gesamtarbeitszeit: 15 Stunden 50 Minuten

Arbeitszeit plausibel, Anfragen möglich:
```

```
Am 01.01.2000 um 00.00 Uhr
     ist niemand im Dienstgebaeude

Am 03.01.2000 um 07.30 Uhr
     betritt       Lehmann              das Dienstgebaeude

Am 03.01.2000 um 07.32 Uhr
     befindet sich Lehmann              im Dienstgebaeude
     betritt       Schulze              das Dienstgebaeude

Am 03.01.2000 um 12.00 Uhr
     befindet sich Lehmann              im Dienstgebaeude
     befindet sich Schulze              im Dienstgebaeude

Am 03.01.2000 um 15.27 Uhr
     befindet sich Lehmann              im Dienstgebaeude
     verlaesst     Schulze              das Dienstgebaeude

Am 03.01.2000 um 15.30 Uhr
     verlaesst     Lehmann              das Dienstgebaeude

Am 04.01.2000 um 07.30 Uhr
     betritt       Lehmann              das Dienstgebaeude

Am 04.01.2000 um 07.32 Uhr
     befindet sich Lehmann              im Dienstgebaeude
     betritt       Schulze              das Dienstgebaeude

Am 04.01.2000 um 12.00 Uhr
     befindet sich Lehmann              im Dienstgebaeude
     befindet sich Schulze              im Dienstgebaeude

Am 04.01.2000 um 15.27 Uhr
     befindet sich Lehmann              im Dienstgebaeude
     verlaesst     Schulze              das Dienstgebaeude

Am 04.01.2000 um 15.30 Uhr
     verlaesst     Lehmann              das Dienstgebaeude

Am 05.01.2000 um 15.30 Uhr
     ist niemand im Dienstgebaeude
```

## Benotung

maximal 25 Punkte

## Lösung:

```
PROGRAM stechuhr ( input, output, passagen ) ;
CONST namenlaenge = 20 ;
TYPE  richtung = ( eintritt, austritt ) ;
      namensform = PACKED ARRAY [ 1..namenlaenge ] OF char ;
      zeitform   = PACKED ARRAY [ 1..12 ] OF char ;
      mp         = ^Mitarbeiter ;
      pp         = ^passage ;
      Mitarbeiter = RECORD
                       name : namensform ;
                       naechster : mp ;
                       erster, letzter : pp
                    END ;
      passage = RECORD
                  zeit : zeitform ;
                  weg : richtung ;
                  naechster : pp
                END ;

VAR   passagen : text ;
      liste, gefunden : mp ;
      name : namensform ;
      zeit : zeitform ;

BEGIN            {   Beginn des Hauptprogramms   }

   reset ( passagen ) ;

   name := '                    ' ;
   liste := erzeugenamen ( nil, name ) ;
   liste^.naechster := liste ;   { leere zyklische Liste erzeugt    }

{   Aufbau der Datenstruktur                                        }

   WHILE   NOT eof ( passagen )   DO BEGIN
      read ( passagen, name ) ;
      gefunden := suchenamen ( liste, name ) ;
      IF   gefunden^.naechster^.name <> name
      THEN   fuegenamenein ( gefunden, name ) ;
      fuegepassagean ( gefunden^.naechster ) ;
      readln ( passagen )
   END ;

{   Ausgabe der Arbeitszeit                                         }

   IF   arbeitszeit ( liste )   THEN   BEGIN

{   Ausgabe der Anwesenheit                                         }

      writeln ( 'Arbeitszeit plausibel, Anfragen möglich:' ) ;
      writeln;
      WHILE   NOT eof ( input )   DO BEGIN
         readln ( zeit ) ;
         anwesend ( liste, zeit )
      END
   END
   ELSE
      writeln ( 'Arbeitszeit nicht plausibel, keine Anfragen möglich !' )

END .
```

# Aufgabe 2

Schreiben Sie die **Funktion** zur Erzeugung eines Knotens für die Personenliste mit Eintragung der Daten für den Namen und die Zeiger, wobei die Schreibweise des Namens und der Zeiger auf den Nachfolger als Eingangsparameter übergeben werden und der Zeiger auf den erzeugten Knoten zurückgegeben wird.

## Benotung

maximal 5 Punkte

## Lösung:

```
{    Erzeugen eines Listeneintrags fuer einen Mitarbeiter          }
{    und Eintragung des Namens und Zeigers auf den naechsten.      }
{    Zurueckgegeben wird der Zeiger auf den erzeugten Eintrag.     }

FUNCTION erzeugenamen ( naechster : mp; name : namensform ) : mp ;
   VAR   neu : mp ;
   BEGIN
      new ( neu ) ;
      neu^.naechster := naechster ;
      neu^.name := name ;
      neu^.erster := nil ;
      neu^.letzter := nil ;
      erzeugenamen := neu
   END ;
```

# Aufgabe 3

Schreiben Sie die **Funktion** zur Erzeugung eines Knotens für die Passagenliste mit Einlesen der Daten und Eintragung in den erzeugten Knoten, wobei der Zeiger auf den erzeugten Knoten zurückgegeben wird.

## Benotung

maximal 8 Punkte

## Lösung:

```
{    Ezeugen eines Listeneintrags fuer eine Passage,               }
{    lesen und eintragen der Daten.                                }
{    Zurueckgegeben wird der Zeiger auf den erzeugten Eintrag.     }

FUNCTION erzeugepassage : pp ;
   VAR   neu : pp ;
         ch : char ;
   BEGIN
      new ( neu ) ;
      neu^.naechster := nil ;
      read ( passagen, neu^.zeit ) ;
      read ( passagen, ch ) ;
      IF  ch = 'E'
      THEN   neu^.weg := eintritt
      ELSE   IF  ch = 'A'
             THEN   neu^.weg := austritt
             ELSE   WRITE ( ' Fehler' ) ;
      erzeugepassage := neu
   END ;
```

## Aufgabe 4

Schreiben Sie die **Funktion** zum Suchen eines Namens in der Personenliste und Rückgabe des Zeigers auf den Vorgänger des vorhandenen Namens oder der Einfügestelle, wobei der Zeiger auf die Personenliste und die Schreibweise des Namens als Eingangsparameter übergeben werden.

**Benotung** maximal 10 Punkte

**Lösung:**

```
{    Suchen eines Namens in der Mitarbeiterliste.                    }
{    Zurueckgegeben wird der Zeiger auf den letzten Eintrag,         }
{    der vor dem gesuchten steht, Zeiger auf den Ursprung,           }
{    wenn kein solcher existiert.                                    }

FUNCTION suchenamen ( liste : mp; name : namensform ) : mp ;
   VAR   lp, sp : mp ;
         stop : boolean ;
   BEGIN
      stop := false ;
      sp := liste ;
      lp := liste^.naechster ;
      WHILE   ( ( lp <> liste )  AND  ( NOT stop ) )   DO BEGIN
         IF   lp^.name >= name   THEN   stop := true
                                 ELSE   BEGIN
                                           sp := lp ;
                                           lp := lp^.naechster
                                 END
      END ;
      suchenamen := sp
   END ;
```

## Aufgabe 5

Schreiben Sie die **Pro e ur** zum Einfügen eines Namens in die Personenliste, wobei der Zeiger auf den Vorgänger und die Schreibweise des Namens als Eingangsparameter übergeben werden.

**Benotung** maximal 5 Punkte

**Lösung:**

```
{    Einfuegen eines neuen Mitarbeiters in die Liste nach dem,       }
{    auf den der Zeiger ep weist.                                    }

PROCEDURE fuegenamenein ( ep : mp; name : namensform ) ;
   BEGIN
      ep^.naechster := erzeugenamen ( ep^.naechster, name )
   END ;
```

## Aufgabe 6

Schreiben Sie die **Pro e ur** zum Anfügen einer Passage an die Passagenliste der referierten Person, wobei der Zeiger auf die Person als Eingangsparameter übergeben wird.

**Benotung** maximal 7 Punkte

### Lösung:

```
{    Anfuegen einer neuen Tuerpassage fuer den durch den            }
{    Zeiger p bezeichneten Mitarbeiter.                             }

PROCEDURE fuegepassagean ( p : mp ) ;
   BEGIN
      IF  p^.letzter = nil
      THEN BEGIN     {    anfuegen der ersten Passage        }
         p^.erster := erzeugepassage ;
         p^.letzter := p^.erster
      END
      ELSE BEGIN     {    anfuegen einer weiteren Passage    }
         p^.letzter^.naechster := erzeugepassage ;
         p^.letzter := p^.letzter^.naechster
      END
   END ;
```

## Aufgabe 7

Schreiben Sie die **Funktion**, die die Arbeitszeit der Mitarbeiter berechnet und ausgibt, sowie die Daten auf Zulässigkeit überprüft. Dabei wird der Zeiger auf die Personenliste als Eingangsparameter übergeben. Im Fehlerfall wird **alse** zurückgeben, sonst **true**.
Stehen in dem **Textfile Passagen** Daten der Form:

```
Schulze            199912301520A
Lehmann            200001030730E
Schulze            200001030732E
Lehmann            200001030830E
Schulze            200001031527A
Lehmann            200001031530A
Lehmann            200001040730E
Schulze            200001040732E
Schulze            200001041527A
Schulze            200001051527A
Lehmann            200001051530A
Lehmann            200001060730E
Schulze            200001060732E
Schulze            200001060727A
```

dann sollte die Ausgabe etwa folgendermaßen aussehen:

```
Lehmann
    03.01.2000 von 07.30 Uhr, Fehler: Eintritt ohne Austritt !
    03.01.2000 von 08.30 Uhr bis 15.30 Uhr: Arbeitszeit 7 Stunden 0 Minuten
    04.01.2000 von 07.30 Uhr bis 15.30 Uhr, Fehler:
        Eintritt und Austritt nicht am selben Tag !
        200001040730 bis 200001051530
    06.01.2000 von 07.30 Uhr, Fehler: Mitarbeiter noch im Betrieb !
    Gesamtarbeitszeit: 7 Stunden 0 Minuten

Schulze
    30.12.1999 von 15.20 Uhr, Fehler: Austritt ohne Eintritt !
    03.01.2000 von 07.32 Uhr bis 15.27 Uhr: Arbeitszeit 7 Stunden 55 Minuten
    04.01.2000 von 07.32 Uhr bis 15.27 Uhr: Arbeitszeit 7 Stunden 55 Minuten
    05.01.2000 von 15.27 Uhr, Fehler: Austritt ohne Eintritt !
    06.01.2000 von 07.32 Uhr bis 07.27 Uhr, Fehler:  Austritt vor Eintritt
        200001060732 bis 200001060727
    Gesamtarbeitszeit: 15 Stunden 50 Minuten

Arbeitszeit nicht plausibel, keine Anfragen möglich !
```

### Benotung

maximal 25 Punkte

## Lösung:

```
{    Ausgabe der Arbeitszeit fuer alle Mitarbeiter.
     Plausibilitätsprüfung :
        Arbeit nicht über Mitternacht
        Eintritt und Austritt abwechselnd
        Beginn mit Eintritt
        Ende mit Austritt
        Eintritt vor Austritt                                  }

FUNCTION arbeitszeit ( liste : mp ) : boolean ;
   VAR   tuer : pp ;
         aktuell : mp ;
         beginn, ende, arbeit, gesamt, i : integer ;
         antritt : zeitform ;
         fehler : boolean ;

   FUNCTION ziffer ( c : char ) : integer ;
      BEGIN
         ziffer := ord ( c ) - ord ( 0 )
      END;

   BEGIN
      arbeitszeit := true ;
      aktuell := liste^.naechster ;
      WHILE   aktuell <> liste   DO   WITH   aktuell^   DO   BEGIN
         writeln ( name )   ;
         gesamt := 0 ;
         tuer := erster ;
         WHILE   ( tuer <> nil )   DO   WITH   tuer^   DO   BEGIN
            write ( '    ', zeit[7], zeit[8], '.',
                            zeit[5], zeit[6], '.',
                            zeit[1], zeit[2], zeit[3], zeit[4],' von ',
                            zeit[9], zeit[10], '.' ,
                            zeit[11], zeit[12], ' Uhr' ) ;
            IF   weg = eintritt
            THEN   BEGIN
               antritt := zeit ;
               beginn := (ziffer(zeit[9])*10+ziffer(zeit[10])) * 60 +
                         ziffer(zeit[11])*10+ziffer(zeit[12]) ;
               IF   naechster <> nil
               THEN   WITH   naechster^   DO   BEGIN
                  IF   ( weg = austritt )
                  THEN   BEGIN
                     write ( ' bis ', zeit[9], zeit[10], '.' ,
                                      zeit[11], zeit[12], ' Uhr' ) ;
                     fehler := false ;
                     FOR i:=1 TO 8 DO
                        fehler := fehler OR ( zeit[i] <> antritt[i] ) ;
                     IF   NOT fehler
                     THEN   BEGIN
                        ende := (ziffer(zeit[9])*10+ziffer(zeit[10])) * 60 +
                                ziffer(zeit[11])*10+ziffer(zeit[12]) ;
                        arbeit := ende - beginn ;
                        IF   arbeit > 0   THEN   BEGIN
                           gesamt := gesamt + arbeit ;
                           writeln ( ': Arbeitszeit ', arbeit DIV 60,
                                     ' Stunden ', arbeit MOD 60, ' Minuten' ) ;
                        END   {   Arbeitszeit eines Tages verbucht  }
                        ELSE   BEGIN
                           arbeitszeit := false ;
                           writeln (', Fehler:  Austritt vor Eintritt' ) ;
                           writeln ( '        ', antritt, ' bis ', zeit )
                        END   {   Austritt vor Eintritt   }
                     END   {   Austritt am selben Tag wie Eintritt   }
                     ELSE   BEGIN
                        arbeitszeit := false ;
```

```
                    writeln (', Fehler:' ) ;
                    writeln ( '        Eintritt und Austritt ',
                             'nicht am selben Tag !' ) ;
                    writeln ( '        ', antritt, ' bis ', zeit )
                 END   {   unzulässige Nachtarbeit   }   ;
                 tuer := naechster
              END   {   Feierabend   }
              ELSE   BEGIN
                 arbeitszeit := false ;
                 writeln (', Fehler: Eintritt ohne Austritt !' ) ;
                 tuer := tuer^.naechster
              END   {   2 Eintritte in Folge   }
           END   {   Passage vorhanden   }
           ELSE   BEGIN
              arbeitszeit := false ;
              writeln (', Fehler: Mitarbeiter noch im Betrieb !' ) ;
              tuer := naechster
           END   {   Mitarbeiter noch im Betrieb   }
        END   {   Dienstbeginn   }
        ELSE   BEGIN
           arbeitszeit := false ;
           writeln (', Fehler: Austritt ohne Eintritt !' ) ;
           tuer := naechster
        END   {   Austritt ohne vorherigen Eintritt   }
     END ;
     writeln ( '    Gesamtarbeitszeit: ', gesamt DIV 60, ' Stunden ',
               gesamt MOD 60, ' Minuten' ) ;
     writeln ;
     aktuell := aktuell^.naechster
  END   {   Arbeitszeitermittlung fuer einen Mitarbeiter   }
END ;
```

## Aufgabe 8

Schreiben Sie die **Pro e ur**, die zu einem gegebenen Zeitpunkt feststellt, welche Person(en) zu diesem Zeitpunkt

- im Betrieb anwesend waren
- gerade zum Dienst erschienen sind
- den Betrieb gerade verlassen haben

oder ob gerade niemand im Betrieb anwesend war. Dabei werden der Zeiger auf die Personenliste und die Zeit als Eingangsparameter übergeben.

### Benotung

maximal 15 Punkte

## Lösung:

```
{    Ausgabe aller zum angegebenen Zeitpunkt im Dienstgebaeude       }
{    anwesenden Mitarbeitern.                                        }

PROCEDURE anwesend ( liste : mp; zeit : zeitform ) ;
   VAR   tuer, letzter : pp ;
         aktuell : mp ;
         stop, da : boolean ;
   BEGIN
      aktuell := liste^.naechster ;
      da := false ;
      writeln ( ' Am ', zeit[7], zeit[8], '.',
                        zeit[5], zeit[6], '.',
                        zeit[1], zeit[2], zeit[3], zeit[4],' um ',
                        zeit[9], zeit[10], '.' ,
                        zeit[11], zeit[12], ' Uhr' ) ;
      WHILE   aktuell <> liste   DO BEGIN
         tuer := aktuell^.erster ;
         letzter := nil ;
         stop := false ;
         WHILE   ( ( tuer <> nil )  AND  ( NOT stop ) )   DO BEGIN
            IF   tuer^.zeit > zeit   THEN   stop := true
                                     ELSE   BEGIN
                                               letzter := tuer ;
                                               tuer := tuer^.naechster
                                     END
         END ;
         IF   letzter <> nil
         THEN   IF   letzter^.zeit = zeit
                THEN   BEGIN
                   IF   letzter^.weg = eintritt
                   THEN   write ( ' betritt       ':20 )
                   ELSE   write ( ' verlaesst     ':20 ) ;
                   writeln ( aktuell^.name, ' das Dienstgebaeude' )   ;
                   da := true
                END
                ELSE   IF   letzter^.weg = eintritt
                       THEN   BEGIN
                          writeln ( ' befindet sich ':20,
                                    aktuell^.name,
                                    ' im Dienstgebaeude' ) ;
                          da := true
                       END ;
         aktuell := aktuell^.naechster
      END ;
      IF   NOT da
      THEN   writeln ( ' ':6, 'ist niemand im Dienstgebaeude' ) ;
      writeln
END ;
```