Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

University of Munich
Institute for Computer Science
Database and Information Systems

# Vorlesung Einführung in die Informatik: Systeme und Anwendungen im SS 2001

# Übungsblatt 2

## Organisatorisches:

**Abgabetermin:** Freitag 1. Juni 2001, 13:00 (strikt)

per Email an die Korrektoren gemäß dem Anfangsbuchstaben desjenigen, der die Email absendet.

| | | |
|---|---|---|
| A – D | funkc@dbs.informatik.uni-muenchen.de | Caroline Funk |
| E – J | martinj@dbs.informatik.uni-muenchen.de | Johanna Martin |
| K – R | reisslh@dbs.informatik.uni-muenchen.de | Harald Reissl |
| S – Z | wiedeman@dbs.informatik.uni-muenchen.de | Florian Wiedemann |

**Wichtig!** Die Email muß des Weiteren folgende Informationen enthalten:
Name, Vorname und Matrikelnummer aller Mitglieder des Teams, das die Aufgabe gemeinsam bearbeitet hat.

# Aufgabe 2 (8 Punkte)

Erstellen sie ein HTML-Dokument, welches die in Abbibldung 1 gezeigte gesichtsartige **Tabelle** ( <TABLE> ) erzeugt.

(Betrachten Sie das als separates HTML-Dokument bereitgestellte Muster der Tabelle aus der Übungsstunde vom 18.5.2001 als Anregung.)

Realisieren Sie die Augen als **Radiobuttons** und den Mund als ein **Eingabefeld**. Beachten Sie, daß die Augen nicht zentriert in ihren Feldern sind, sondern näher bei der Nase.

(<form action ...> nicht vergessen! z.B.   zur Erzeugung leerer Felder verwenden.)

**Zu Beachten:** Stellen Sie **entweder** sicher, daß Ihr HTML-Dokument mit Netscape 4.7 ladbar ist, **oder** schicken Sie zusätzlich (in der selben Mail) einen Postscript oder PDF Save des geladenen HTML-Dokuments. (Auf diesen "Postscript oder PDF Saves" können gewisse Dinge fehlen (z.B. Radiobuttons). Falls diese im HTML-Dokument vorhanden sind, ist das kein Problem.)

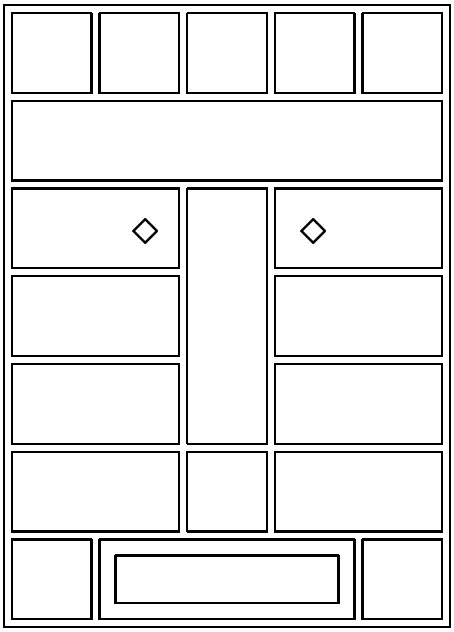

Abbildung 1: gesichtsartige Tabelle

**WARNUNG:** Lösen Sie HTML-Aufgaben mit einem normalen Editor und nicht mit speziellen HTML-Editoren, Composern, etc. da Ihnen diese Programme in der Klausur auch nicht zur Verfügung stehen.

# Aufgabe 3 (insgesamt 20 Punkte)

**WICHTIG:** Schicken Sie als Lösung dieser Aufgabe reine ASCII-Dateien – keine Word-Documente, Excel-Dateien o.ä.
Verwenden Sie keine Tabulatorbefehle oder Proportiaonal-Fonts — speziell in den zu erstellenden Ablaufprotokollen — da dies bei anderen Tabulatoreinstellungen oder anderen Fonts zu großen Verzerrungen führen kann, was die Lesbarkeit stark erschweren oder ganz unmöglich machen kann.
**Unleserlichkeit führt zu Punkteabzug!**

Im Folgenden sind 3 Varianten des Vorschlags von Hyman — siehe Anhang: Wechselseitiger Ausschluß — zur Realisierung des wechselseitigen Ausschlusses der kritischen Bereiche zweier Prozesse gegeben.

Die Programme aller 3 Varianten benützen folgendes Array:

```
b:  ARRAY[1..2] of BOOLEAN := (false,false);
```

## Variante 1 (6 Punkte)

"Versuch mit einer Implementierung ohne Ticket."

```
Prozess P1 :                          Prozess P2:

BEGIN                                 BEGIN
    WHILE b(2) DO;                        WHILE b(1) DO;
    b(1):=true;                           b(2):=true;
    -- kritischer Bereich                 -- kritischer Bereich
    b(1):=false;                          b(2):=false;
END;                                  END;
```

Erläuterung:
`WHILE b(2) DO;` bedeutet warten bis `b(2)` falsch wird.

⇒ Zeigen Sie an Hand eines Ablaufprotokolls, daß es möglich ist, daß die Prozesse $P_1$ und $P_2$ gleichzeitig in den kritischen Bereich eintreten.

## Variante 2 (8 Punkte)

"Erst setzen, dann testen"

```
Prozess P1 :                          Prozess P2:

BEGIN                                 BEGIN
    b(1):=true;                           b(2):=true;
    WHILE b(2) DO;                        WHILE b(1) DO;
    -- kritischer Bereich                 -- kritischer Bereich
    b(1):=false;                          b(2):=false;
END;                                  END;
```

⇒ Begründen Sie, daß maximal ein Prozeß gleichzeitig in den kritischen Bereich eintreten kann.

⇒ Zeigen Sie an Hand eines Ablaufprotokolls, daß es möglich ist, daß sich die Prozesse $P_1$ und $P_2$ verklemmen können.

## Variante 3 (6 Punkte)

"Zurücknehmen des eigenen Wunsches, wenn der andere Prozeß seinen Wunsch bereits geäußert hat."

```
Prozess P1 :

BEGIN
    b(1):=true;
    WHILE b(2) DO
        BEGIN
            b(1):=false;
            -- warten
            b(1):=true;
        END;
    -- kritischer Bereich
    b(1):=false;
END;
```

```
Prozess P2:

BEGIN
    b(2):=true;
    WHILE b(1) DO
        BEGIN
            b(2):=false;
            -- warten
            b(2):=true;
        END;
    -- kritischer Bereich
    b(2):=false;
END;
```

Erläuterungen:

`WHILE b(2) DO BEGIN ...END;` : Die Anweisungen zwischen `BEGIN ...END;` werden solange ausgeführt solange `b(2)` wahr ist.

'`-- warten`' kann als aktives Warten verstanden werden, d.h. es wird nicht automatisch der Prozessor abgegeben und der Prozess deaktiviert.

⇒ Zeigen Sie an Hand eines Ablaufprotokolls, daß es möglich ist, daß die Prozesse $P_1$ und $P_2$ beide verhungern.

# Anhang: Wechselseitiger Ausschluß

(wird am 25.5.2001 in der Übung besprochen.)

Um "sinnvolle" Berechnungen von nebenläufigen Prozessen eines Rechensystems zu gewährleisten, sind Maßnahmen zur Vermeidung von Konflikten, die durch den konkurrierenden Zugriff auf gemeinsam benutzte Datenobjekte entstehen, zu ergreifen.

H. Hyman schlug 1966 folgende Lösung für den wechselseitigen Ausschluß der kritischen Bereiche zweier Prozesse $P_1$(i=1) und $P_2$(i=2) vor:

```
 1  b:  array(1..2) of boolean := (false, false);
 2  k:  integer := 1;
 3  BEGIN
 4  C0:   b(i) := true;
 5        IF k <> i THEN
 6  C1:      IF b(3-i)
 7             THEN GOTO C1
 8             ELSE k:=i
 9           ENDIF;
10        ENDIF;
⋮         ....... --- kritischer Bereich;
11+n      b(i) := false;
.         ....... --- Rest des Programms;
.         GOTO C0;
.   END;
```

Die Hilfskomponenten sind hier die Variablen $b$ und $k$ mit den auf ihnen definierten Lese- und Schreiboperationen. Die Variablen $b$ und $k$ seien wie angegeben initialisiert.

- Die Variable $k$ kann man als eine Art Ticket interpretieren, das abwechselnd den Prozessen zugeteilt wird und zum Eintritt in deren kritischen Bereich berechtigt.
- Über die Variable $b$ kann ein Prozeß testen, ob der andere Prozeß seine Zugangserlaubnis zum kritischen Bereich in Anspruch nehmen will oder nicht.

Erläuterungen zu obigem Programm:

1. Zeile 1: weder Prozess $P_1$ noch $P_2$ hat den Wunsch in den kritischen Bereich einzutreten
2. Zeile 2: Prozess $P_1$ hat Berechtigung in den kritischen Bereich einzutreten.
3. Zeile 3: Hier beginnt Programm. Davor nur allg. Initialisierung.
4. Zeile 4: Prozess $P_i$ äußert den Wunsch in den kritischen Bereich einzutreten
5. Zeile 5: `k <> i` : Prozess $P_i$ hat noch keine Berechtigung

6. Zeile 6: `b(3-i)` : Wunsch des anderen Prozesse in kritischen Bereich einzutreten.

7. Zeile 8: anderen Prozess will nicht in kritischen Bereich einzutreten, Prozess $P_i$ darf: `k := i`

8. Zeile 11+n: Prozess $P_i$ löscht seinen Wunsch

Diese angebliche Lösung für den wechselseitigen Ausschluß der kritischen Bereiche zweier Prozesse läßt sich durch folgendes Ablaufprotokoll widerlegen, welches zeigt daß es eine Möglichkeit gibt, in der beide Prozesse gleichzeitig ihre kritischen Bereiche ausführen. (Dabei wird davon ausgegangen, daß die beiden Prozesse das angegebene Programm nebenläufig ausführen.)

**Ablaufprotokoll**
Prozessorwechsel wird durch die horizontalen Linien angezeigt.

| $P_1$ (i=1) | $P_2$ (i=2) | Werte |
|---|---|---|
| | | k=1<br>b(1)=b(2)=false |
| | b(2) := true;<br>IF k <> 2 (=*true*) THEN<br>IF b(1) (=*false*) | b(2)=true<br>k=1<br>b(1)=false |
| b(1) := true;<br>IF k <> 1 (=*false*) THEN<br>...<br>-- kritischer Bereich | | b(1)=b(2)=true<br>k=1 |
| | ELSE k := 2;<br>...<br>-- kritischer Bereich | k=2<br><br>b(1)=b(2)=true |

Erläuterungen zum Ablaufprotokoll:

Prozess $P_2$ unterbricht die Programmabarbeitung nach Bearbeitung von Zeile 6 — Zeile 7 wird übersprungen — und es geht erst weiter nachdem Prozess $P_1$ bei der Abarbeitung des Programms in den kritische Bereich eingetreten ist.