Professor Dr. F. Schweiggert    Abteilung Angewandte Informationsverarbeitung    29. Januar 2001
Johannes Mayer 

# Allg. Informatik für WiWi (WS 2000/2001)

**Abgabetermin: 8. Februar 2001**

## Beispiellösung

## 1 Arbeiten mit Bildern (30 Punkte)

Sie dürfen nun etwas ganz anschauliches machen. Sie bekommen Schwarz-Weiss-Bilder im PBM-Format und sollen diese einlesen, verändern und ausgeben. Das Schöne an dieser Aufgabe ist, dass sie sich die Programm-Ausgabe als Bild betrachten können.
Zunächst ein paar Worte zum PBM-Format. (Wir betrachten hier nur die ASCII-Version, d. h. sie können davon ausgehen, dass es sich bei dem Bild um eine Textdatei handelt.) Das Bild beginnt mit der typischen Zeichenkette „`P1`". Danach folgen ein (oder mehrere) Whitespace(s), d. h. Leerzeichen, Tabulatoren oder Zeilenumbrüche. Danach kommen die Breite des Bildes, Whitespaces, die Höhe des Bildes und wieder Whitespaces. Danach kommt für jeden Punkt des Bildes genau eine Zahl (0 oder 1), wobei zunächst die Punkte der ersten Zeile von links nach rechts, dann die der zweiten und so weiter kommen und die Zahlen durch Whitespace(s) voneinander getrennt sind. Die 1 repräsentiert dabei einen schwarzen und die 0 einen weissen Punkt. Sie können davon ausgehen, dass die Bilder genau so aufgebaut sind und die Anzahl der Punkte gleich Höhe mal Breite ist. Zudem sollte jede Zeile höchstens 70 Zeichen enthalten, was sie bei der Ausgabe beachten müssen. (Für weitere Informationen siehe `man pbm`.) Beispiel:

```
P1
5 5
0 1 1 0 0
0 1 0 1 1
1 0 0 1 1
1 1 1 0 0
1 1 1 1 1
```

Zur internen Verwaltung eines Bildes ist ihnen die folgende Datenstruktur vorgegeben:

```
CONST MAXWIDTH = 1000;
      MAXHEIGHT = 800;

TYPE PBMImage = RECORD
```

```
          width, height : INTEGER;
          pixel : ARRAY MAXWIDTH, MAXHEIGHT OF BOOLEAN;
       END;
```

Wir gehen also davon aus, dass ein Bild nicht größer als 1000 mal 800 ist. Im Feld width ist die Breite und in height die Höhe des Bildes in Pixeln enthalten. Das Array pixel enthält für jeden Punkt des Bildes einen Eintrag. Ein schwarzer Punkt soll in diesem Array als Wert `TRUE` abgespeichert werden (somit repräsentiert also `FALSE` einen weissen Punkt).
Schreiben sie nun ein Oberon-Programm mit den im folgenden beschriebenen Prozeduren. Zuächst wollen wir von der Standardeingabe ein Bild im PBM-Format einlesen. Dazu schreiben wir die Prozedur

```
PROCEDURE ReadPBM(VAR image: PBMImage);
```

die dann das eingelesene Bild im VAR-Parameter `image` zurückgibt. Außerdem benötigen wir zur Ausgabe von `image` auf die Standardausgabe die folgende Prozedur:

```
PROCEDURE WritePBM(image: PBMImage);
```

Jetzt möchten wir natürlich auch am Bild etwas verändern. Also schreiben wir die Prozedur

```
PROCEDURE InvertPBM(VAR image: PBMImage);
```

mit der die Farben im Bild vertauscht werden (aus einem schwarzen Punkt wird ein weisser und umgekehrt). Zu guter Letzt möchten wir das Bild noch horizontal spiegeln (d. h. erste und letzte Zeile vertauschen, zweite und vorletzte Zeile vertauschen, ..., aber nur bis zur Mitte). Dazu schreiben wir folgende Prozedur:

```
PROCEDURE HReflectPBM(VAR image: PBMImage);
```

Im Hauptprogramm sollen alle Prozeduren, die Sie geschrieben haben, verwendet werden. Testen können Sie Ihr Programm mit den Bildern *unilogo.pbm*, *trophy.pbm* und *2dimnor.pbm*.
(Hinweise: Angenommen Ihr Programm heißt `PBM`, dann können sie mit der Kommandozeile `PBM < trophy.pbm | xv -` die Ausgabe Ihres Programms mit dem Programm `xv` als Bild betrachten. Beim PBM-Format sind auch Kommentarzeilen, die mit `#` beginnen, zugelassen. Diese ignorieren wir hier der Einfachheit halber.)

**Lösung:**

```
DEFINITION PBM;
END PBM.
```

---

```
MODULE PBM;

   IMPORT Read, Write;

   CONST MAXWIDTH = 1000;
         MAXHEIGHT = 800;
```

```
(* Datenstruktur fuer ein Bild im PBM-Format:
 *   width       = Breite des Bildes in Punkten
 *   height      = Hoehe des Bildes in Punkten
 *   pixel[x][y] = true, falls Punkt bei (x,y) schwarz
 *                 wobei 0 <= x < width und 0 <= y < height
 *)
TYPE PBMImage = RECORD
                   width, height : INTEGER;
                   pixel : ARRAY MAXWIDTH, MAXHEIGHT OF BOOLEAN;
                END;

(* Bild-Variable fuer das Hauptprogramm *)
VAR picture : PBMImage;

(* Einlesen eines Bildes im PBM-Format von der
 * Standardeingabe und Ablegen in image
 *)
PROCEDURE ReadPBM(VAR image: PBMImage);
   VAR tag : ARRAY 3 OF CHAR;    (* zum einlesen von "P1" *)
       x, y, p : INTEGER;
BEGIN
   Read.String(tag);             (* bei korrektem Bild immer "P1" *)
   Read.Int(image.width);
   Read.Int(image.height);

   y := 0;
   WHILE y < image.height DO
      x := 0;
      WHILE x < image.width DO

         Read.Int(p);            (* lese Punkt (1 = schwarz, 0 = weiss) *)
         image.pixel[x][y] := (p = 1);  (* true, falls Punkt schwarz *)

         INC(x);
      END;
      INC(y);
   END;
END ReadPBM;

(* Ausgabe des Bildes image im PBM-Format
 * auf die Standardausgabe
 *)
PROCEDURE WritePBM(image: PBMImage);
   VAR x, y, lineLength : INTEGER;
BEGIN
```

```
   Write.String("P1");              (* Kennung eines ASCII-PBM-Bildes *)
   Write.Ln;

   Write.Int(image.width, 0);
   Write.Char(" ");
   Write.Int(image.height, 0);
   Write.Ln;

   lineLength := 0;

   y := 0;
   WHILE y < image.height DO
      x := 0;
      WHILE x < image.width DO

         IF image.pixel[x][y] THEN  (* falls true, dass 1 fuer schwarz *)
            Write.String("1 ");
         ELSE                       (* sonst 0 fuer weiss *)
            Write.String("0 ");
         END;

         INC(lineLength, 2);        (* Laenge der aktuellen Zeile *)
         IF lineLength >= 70 THEN   (* 70 oder mehr Zeichen => Umbruch *)
            Write.Ln;
            lineLength := 0;
         END;

         INC(x);
      END;
      INC(y);
   END;
END WritePBM;

(* Bild image invertieren, d.h. aus schwarzem Punkt wird
 * weisser Punkt und umgekehrt
 *)
PROCEDURE InvertPBM(VAR image: PBMImage);
   VAR x, y : INTEGER;
BEGIN
   y := 0;
   WHILE y < image.height DO
      x := 0;
      WHILE x < image.width DO

         (* aus true wird false und umgekehrt => Farbvertauschung *)
         image.pixel[x][y] := ~ image.pixel[x][y];
```

```
        INC(x);
      END;
      INC(y);
   END;
END InvertPBM;

(* Inhalte der beiden booleschen Variablen
 * a und b vertauschen
 *)
PROCEDURE Swap(VAR a, b: BOOLEAN);
   VAR tmp : BOOLEAN;
BEGIN
   tmp := a;
   a := b;
   b := tmp;
END Swap;

(* Bild image horizontal spiegeln, d.h. erste und letzte
 * Zeile vertauschen, zweite und vorletzte Zeile vertauschen
 * usw. aber nur bis zur Mitte, da sonst wieder das urspruengliche
 * Bild produziert wird !
 *)
PROCEDURE HReflectPBM(VAR image: PBMImage);
   VAR x, y : INTEGER;
BEGIN
   y := 0;
   (* nur solange die y-ten Zeile von oben einen kleineren Index
    * hat wie die y-ten Zeile von unten => nur bis zur Mitte vertauschen
    * => andernfalls wuerde das urspruengliche Bild rekonstruiert
    *   y                    = Index der y-ten Zeile von oben
    *   image.height - 1 - y = Index der y-ten Zeile von unten
    *)
   WHILE y < (image.height - 1) - y DO
      x := 0;
      WHILE x < image.width DO

         (* vertausche je zwei korrespondierende Punkte (d.h.
          * Punkte mit derselben x-Koordinate) der beiden
          * Zeilen
          *)
         Swap(image.pixel[x][y], image.pixel[x][image.height - 1 - y]);

         INC(x);
      END;
      INC(y);
```

```
      END;
   END HReflectPBM;

BEGIN
   ReadPBM(picture);       (* PBM-Bild von der Standardeingabe lesen *)
   InvertPBM(picture);     (* Bild danach invertieren *)
   HReflectPBM(picture);   (* Bild horizontal spiegeln *)
   WritePBM(picture);      (* PBM-Bild auf Standardausgabe schreiben *)
END PBM.
```