

## – Joe's Problem

Ausarbeitung von Bernd Ahrens und Mark Laumann  
22.01.2003

Als erstes wurde Joe's Problem zu unserem ... hierbei ging es um Joe ....

Joe gehört nämlich zu einem Geheimbund. Dieser besteht aus 40 Personen und sie wollen einen neuen geheimen Obergeheimler bestimmen. Dazu stellen sie sich im Kreis auf. Beim alten Obergeheimler beginnen sie mit 1 auszuführen. Wer bei 2 ankommt, scheidet aus und der nächste beginnt wieder mit 1. Wo müsste Joe sich also hinstellen, um neuer Obergeheimler zu werden.

### – erste Prozedur (A)

Die ERSTE Prozedur

... nen Haufen Arbeit ... an sich - recht solide ... bei Zahlen jenseits von 2 Bildschirmängen aber doch schon recht langsam ...

Die vielen internen (aber mittlerweile abgeschalteten) print-Befehle dienten dem Auffinden von Fehlern, bzw. um herauszufinden an welcher Stelle sich die Prozedur aufgehängt hat.

Zur Funktionsweise sei noch kurz angemerkt, daß die Prozedur im Grunde aus 2 Teilen besteht.

Dem Ersten und dem Zweiten.

Nun zum Ersten.

Dieser Teil rechnet die Anfangszahl so lange herunter (bzw. streicht so lange jede 2te Zahl) bis nur noch 1 Element (Zahl) übrig bleibt.

```
> A:=proc(x) local a,b,g,f,t,h:
>   if x < 1 then print ("So läuft das hier nicht, mein Lieber
!"):
>   print ("Such dir ne andere Zahl aus, aber diesmal bitte
GRÖßER als NULL"):
>   else:
>
>   g[1]:=U:           Hier wird die "1" (Anfang erste Runde) als "U"ngerade
definiert
>   a[1]:=x:          Hier wird die Anzahl der Elemente (Personen) der ersten
Runde über Eingabe"x" definiert.
>   for b do:         Diese Schleife ist für die Runden "b" zuständig.
>
>     if g[b] = U then
>       if gcd(2,a[b]) = 2 then a[b+1]:=(a[b])/2; g[b+1]:=U;
#print(b,a[b],g[b],"kVW",a[b+1]);
>       else
>         a[b+1]:=((a[b])+1)/2;
g[b+1]:=G;#print(b,a[b],g[b],"VW",a[b+1]);
>       end if:
>       if a[b]=1 then break; end if;           VW:=V
orzeichenWechsel
>
>     else
>       if gcd(2,a[b]) = 2 then a[b+1]:=(a[b])/2; g[b+1]:=G;
```

```

    #print(b,a[b],g[b],"kVW",a[b+1]);
>     else                                a[b+1]:=((a[b])-1)/2;
g[b+1]:=U;#print(b,a[b],g[b],"VW",a[b+1]);
>     end if;
>     if a[b]=1 then break;end if;
>     end if;
>
> end do:
> end if:
>

```

Der 2te Teil, der übrigens hier beginnt, rechnet dann anhand der im 1. Teil gewonnenen Daten die Position zurück, bis zur Platznummer auf der man nachher zu stehen hat. Erklärungen zur Funktion dieses Prinzips stehen im unteren Teil des Worksheet unter dem Punkt "Zu den Aufgaben".

```

> h[b-1]:=1;                                "h" beschreibt in
    Abhängigkeit von "b" die Platznummer
> for t from b-1 to 1 by -1 do;
>   if g[t]=U then h[t-1]:=(h[t]*2)-1;
    #print(t,"ungerade",h[t-1]);
>   else h[t-1]:=(h[t])*2;
    #print(t,"gerade",h[t-1]);
>   end if;
> end do;
>
> #print (b, Runden, Platznummer ,h[0]);
> h[0];
> end proc:

```

## – Formel (F)

Die Formel ist eigentlich ein Zufallsprodukt. Sie entstand durch Vergleichen der Ergebnisse ersten (ca. 20) manuell erstellten Abzählreihen (siehe "Plot"). Ein DIREKTER mathematischer Zusammenhang ist also nicht zu erwarten ... aber sie läuft ...

```

> F:=proc(x) local m,M,n,N,g,G,f;
>   if x < 1 then print ("So läuft das hier nicht, mein Lieber
    !");
>   print ("Such dir ne andere Zahl aus, aber diesmal bitte
    GRÖßER als NULL");
>   else
>     n:= log [2](x);
>     N:= simplify (n);
>     m:= floor(n)+1;
>     M:= simplify (m);
>     f:= (-1)*(2^(M)-1-(2*x));
>
>                                     #print (M(x),f(x));
>   end if;
> end proc:

```

## – Abstandmessung

Mit diesem kleinen Programm kann überprüft werden (bzw. wurde überprüft), ob es bei der Berechnung der Platznummer Unterschiede zwischen der Formel und der erste Prozedur gab. ... Es gab keine .... also weiter im Text ...

```

> J:=proc(x,y) local i,f,L,T;
>   for i from x to y do;
>     f[x-1]:=0;
>     L[i]:=A(i);
>     T[i]:=F(i);
>     f[i]:=T[i]-L[i]+f[i-1];
>
>   end do;
>   print (f[i-1]);
> end proc:

```

## – Plot

Um dem Ganzen ein wenig optische Unterstützung zu geben, wurde hier die oben angegebene Formel für die ersten 257 Elemente graphisch dargestellt. Sie ist im weiteren eigentlich total überflüssig.

```

[ > plot(F,1..257,adaptive=false,numpoints=257,style=POINT);

```

## – verbesserte Prozedur (B)

Bei dieser Prozedur ist im Gegensatz zur letzten die ggT-Funktion gegen eine einfachere Modulo-Funktion ersetzt worden (was doch erheblich an Zeit spart, wie man bei Punkt "ZEITMESSUNG" bald feststellen wird). Desweiteren wurde "x" als Eingabe auf ganze Zahlen festgelegt. Unter anderem ist an der Übersichtlichkeit gearbeitet worden.

(Kommentare haben wir uns gespart, da die Prozedur nicht wesentlich verändert wurde)

```

> B:=proc(x::integer) local a,b,g,f,i,t,h;
>   if x < 1 then print ("So läuft das hier nicht, mein Lieber
!"):
>   print ("Such dir ne andere Zahl aus, aber diesmal bitte
GRÖßER als NULL"):
>   else:
>   i:=0: g[1]:=U: a[1]:=x:
>
>   for b do:
>
>     if g[b] = U then
>       if modp(a[b],2)=0 then a[b+1]:=(a[b])/2;
g[b+1]:=U;#print(b,a[b],g[b],"kVW",a[b+1]);
>       else
a[b+1]:=((a[b])+1)/2;g[b+1]:=G;#print(b,a[b],g[b],"VW",a[b
+1]);
>       end if:
>     else
>       if modp(a[b],2)=0 then a[b+1]:=(a[b])/2;
g[b+1]:=G;#print(b,a[b],g[b],"kVW",a[b+1]);
>       else
a[b+1]:=((a[b])-1)/2;g[b+1]:=U;#print(b,a[b],g[b],"VW",a[b
+1]);
>       end if:
>     end if;
>     if a[b]=1 then break;end if;
>   end do:
> end if:
>

```

```

> h[b-1]:=1;
> for t from b-1 to 1 by -1 do;
>   if g[t]=U then h[t-1]:=(h[t]*2)-1;
>   #print(t,"ungerade",h[t-1]);
>   else           h[t-1]:= h[t]*2;
>   #print(t,"gerade",h[t-1]);
>   end if;
> end do;
> h[0];
> #print (b,Runden,Platznummer,h[0]);
> end proc;

```

## – Zeitmessung

Hier haben wir eine einfache Zeitmessung durchgeführt. Dabei sind die Zeiten von Oben nach Unten entsprechend der Entwicklung der Algorithmen aufgelistet. Oben steht die Zeit der erten Prozedur, die noch mit dem ggT arbeitet, man sieht darunter die deutliche Zeitersparnis der verbesserten Prozedur. Als letztes ist die Zeit der Formel angegeben. Über den 3 Zeiten steht überigens NUR die Platznummer, nicht die Anzahl der Elemente (Personen).

```

> st:=time():
> A(13001^502);
> time()-st;
> st:=time():
> B(13001^502):
> time()-st;
> st:=time():
> F(13001^502):
> time()-st;

```

## – verbesserte allgemeine Prozedur (C)

Das Non-plus-Ultra schlecht hin ... (<- These !)

Optisch rausgeputzt (sogar die local-begrenzten Variablen sind alphabetisch geordnet, wow)

Alte Print-Befehle sind trotzdem noch in dieser Prozedur integriert, um eine bessere Anschauung der ablaufenden Prozesse zu ermöglichen.

Unter anderem ist die Bedienung "freundlicher" geworden. (Kommentare und Anleitung zu dieser Prozedur stehen unter "Zu den Aufgaben - A6.2")

```

> C:=proc(x::integer,y::integer) local a,b,f,g,h,i,k,t,z:
>   if x<2 or y<2 then print ("Es wäre empfehlenswert, eine
>   Zahl größer als 1 zu wählen"):
>   print ("27 und 3 sollen schöner sein - hat mal wer
>   behauptet"):
>   else:
>   a[1]:=x: g[1]:=1: i:=0:
>
>   for b do:
>     z[b]:= modp(g[b],y);
>     k[b]:= modp(a[b],y);
>
>     if z[b] = 0           then a[b+1]:=a[b]- (a[b] +
>     modp(y-k[b],y))/y;
>     elif z[b]+k[b] > y then a[b+1]:=a[b]- (a[b] +

```

```

    y-k[b])/y;                                #1
>     else                                     a[b+1]:=a[b]- (a[b] - k[b])
    /y;
>     end if;
>     g[b+1]:=modp(g[b]+k[b],y);
    #print(b,'z'=z[b],'k'=k[b],'a'=a[b],'g'=g[b]);
>
>     if a[b]=1 then break;end if;
>     end do;
>     end if;
>
> h[b-1]:=1;
> for t from b-1 to 1 by -1 do;
    #2
>     if g[t]=0 then h[t-1]:=trunc((h[t]-2+y)/(y-1))+h[t];
    #print(,g[t],h[t],h[t-1]);
>     else
    h[t-1]:=trunc((h[t]-2+modp(g[t],y))/(y-1))+h[t];
    #print(,g[t],h[t],h[t-1]);
>     end if;
>     end do;
> h[0];
    #print (b,Runden,Platznummer,h[0]);
> end proc:

```

## – einige Beispiele aus dem alltäglichen Leben

Würde man den Bundeskanzler nicht mehr wählen, sondern über die 2er-Regel auszählen, sollte man bei einer aktuellen Bevölkerungszahl von 82.518.379 Einwohner (Stand 3.Quartal 2002), auf folgendem Platz stehen. `F(82518379);`

oder würde die momentane Weltbevölkerung ausgezählt, sollt man sich hier plazieren. `F(4356695690);`

Für den Fall, daß Gevatter Tod nach selbigem System vorgeht, und er es schafft, pro 1/4 Sekunde einen Menschen abzuzählen (zu streichen), sprich in jeder Sekunden 2 Menschen sterben müssen, und man vorraussetzten würde, daß es keine Geburten mehr geben würde, benötigte er nur schlappe 69,2 Jahre um die Menschheit zu eliminieren.

`evalf(4356695690/2/60/60/24/364);`

## – Abzählreime

Für die folgenden Abzählreime wurde für Gruppen bis zu 10 Personen, die entsprechend gewinnende Platznummer ermittelt:

Ene mene Miste, es rappelt in der Kiste, ene mene Meck, und du bist weg (y = 15)

```
> C(2,15),C(3,15),C(4,15),C(5,15),C(6,15),C(7,15),C(8,15),
    C(9,15),C(10,15);
```

Drei Polizisten pissten in die Kisten, einer pisst vorbei, du bist frei! (y=12)

```
> C(2,12),C(3,12),C(4,12),C(5,12),C(6,12),C(7,12),C(8,12),
    C(9,12),C(10,12);
```

Blöder Hund, blode Kuh, rein bin ich - raus bist du! (y=10)

```
> C(2,10),C(3,10),C(4,10),C(5,10),C(6,10),C(7,10),C(8,10),
    C(9,10),C(10,10);
```

Kleiner Schneck, du musst weg! (y=5)

```
> C(2,5), C(3,5), C(4,5), C(5,5), C(6,5), C(7,5), C(8,5),
    C(9,5), C(10,5);
```

## – Zeitmessung (die 2te)

Hier soll erneut die Schnelligkeit der einzelnen Prozeduren ermittelt werden. Man sieht, dass die Prozedur C zwar um einiges langsamer im Gegensatz zu ihrem Vorgänger ist, dafür kann sie aber auch mehr leisten. Immer noch deutlich schnellste ist die Formel von ganz oben, die nur einen Bruchteil der Zeit braucht. Die Zahl der Elemente (bzw. Personen) hat lediglich 2066 Stellen, ist also ein bisschen GRÖßER.

```
> st:=time():
A(13001^502):
  A,time()-st;
st:=time():
B(13001^502):
  B,time()-st;
st:=time():
C(13001^502,2):
  C,time()-st;
st:=time():
F(13001^502):
  F,time()-st;
F(13001^502);
```

## – Proben von Funktionen

[ Hier wurden Funktionen wie "modp" und "trunc" auf ihre Funktionalität geprüft.

```
> a:=16;y:=3;g:=0;
> modp(a,y);modp(g,y);print ('a'=a,'y'=y);
> trunc(2);
```

## – Test für gerade/ ungerade

[ Dies war lediglich nur zur Überprüfung, ob das Programm gerade und ungerade Zahlen unterscheiden kann.

```
> for n from 0 to 10 do;
> if gcd(2,n)=2 then print (n,"g");
> else print (n,"U");
end if;
end do;
```

[ Für die erste Wahl mit 40 Personen sollte sich Joe auf folgenden Platz stellen:

```
> A(40);
```

[ Für die Wahl im 2. Jahr wäre folgender Platz empfehlenswert:

```
> C(40,7);
```

[ Weitere Beispiele, zum selber ausprobieren ...

```
> F(12);
```

```
> C(27,3);
```

```
> C( , );
```

```
> C( , );
```

```
> J(1,50 );
```

## – Zu den Aufgaben

### – A 6.1

– (i)

Übertrage das Modell in den Rechner und berechne einige Lösungen kleiner Mengen ( $n = 1$  bis  $n = 20$ ). Was kannst Du über große Mengen

(z.B.  $n = 2^{20}$ ) sagen?

> A(1), A(2), A(3), A(4), A(5), A(6), A(7), A(8), A(9), A(10);  
A(11), A(12), A(13), A(14), A(15), A(16), A(17), A(18), A(19), A(20);

Bei sämtliche  $2^n$  (für  $n$  beliebig aus  $\mathbb{N}$ ) ist die "gewinnende" Platznummer immer die Nummer 1. Also bleibt der Obergeheimler bestehen.

> A( $2^{20}$ ), A( $2^5$ ), A( $2^{11}$ ), A( $2^{33}$ ), A( $2^{87}$ ), A( $2^{14}$ ), A( $2^8$ ), A( $2^4$ ), A( $2^{136}$ ), A( $2^{17}$ );

(ii)

Was genau passiert in einer Runde?

In einer Runde werden alle Elemente herausgeworfen, die eine gerade Platznummer haben. Dies betrifft natürlich auch (spätestens nach der ersten Runde) ungerade Zahlen die eine gerade Platznummer zugewiesen bekommen haben..

(iii)

Welche Elemente haben welche Platznummer?

Die Elemente die in dieser Runde rausfliegen hatten die geraden Platznummern, jene welche die Runde überdauern, hatten die Ungeraden inne.

(iv)

Wovon hängt das ab?

Das hängt von der letzten Runde ab. Wenn in der letzten Runde eine gerade Anzahl von Elementen vorhanden war, so ist genau die Hälfte "entsorgt" worden. Dies hat zur Folge, daß das erste Element in dieser Runde den gleichen Rest (betrachtet man das Element modulo 2) hat wie das erste Element der letzten Runde. Im Gegensatz dazu ändert sich der Rest des ersten Elementes der aktuellen Runde, wenn die vorhergehende Runde eine ungerade Anzahl von Elementen hatte. (Als graphische Unterstützung wird hier die Excel-Tabelle "Joe's Problem - erste Tabelle" empfohlen, sie ist ebenfalls als Unterstützung beim nächsten Punkt (v) gedacht, und erklärt kurz und knapp obwohl mehr knapp als kurz - graphisch (und mit Farben untermalt) die Zusammenhänge beim "Runterzählen", sowie das System des "Raufzählens" um die entsprechende Platznummer ermitteln zu können)

(v)

Erstelle aus der Fallunterscheidung eine rekursive Funktion zur Berechnung des letzten Elementes. Was kannst Du über die Brauchbarkeit sagen?

Bei der 2er-Abzählung unterscheidet man 4 Fälle:

1. Fall: vorherige Reihe beginnt mit gerader Zahl und hat gerade Anzahl von Elementen => diese beginnt mit gerader Zahl (Hälfte ist rausgeflogen)

2. Fall: vorherige Reihe beginnt mit ungerader Zahl und hat gerade Anzahl von Elementen => diese beginnt mit ungerader Zahl (Hälfte ist rausgeflogen)

3. Fall: vorherige Reihe beginnt mit gerader Zahl und hat ungerade Anzahl von Elementen => diese beginnt mit ungerader Zahl (Hälfte + 1 ist raus)

4. Fall: vorherige Reihe beginnt mit ungerader Zahl und hat ungerade Anzahl von Elementen => diese beginnt mit ungerader Zahl (Hälfte - 1 ist raus)

Anhand der Tatsache, daß jedes mal die Hälfte ( ) rausfliegt, und dass in einer Reihe, welche mit einer ungeraden Zahl beginnt, man niemals eine gerade Positionsnummer haben darf, und umgekehrt, kommt man auf folgende Regel zur Errechnung der Positionsnummer anfangen in der "letzten Runde":

In der letzten Runde ist man das einzigste Element, man besitzt also die Positionsnummer 1. Ist nun die vorherige Runde mit einer geraden Zahl begonnen

worden, so darf man auf nicht ungeraden Positionen sein, man multipliziere also seine Positionsnummer einfach mit 2. Begann die vorherige Reihe aber mit einer ungeraden Zahl, so darf man sich nur auf ungeraden Platznummern tummeln, man multipliziere dito seine Positionsnummer mit 2 und ziehe 1 ab. So verfährt man bis man sich in der letzten Runde befindet - und siehe da - man erhält die Positionsnummer, die den "Sieger" dieser Auszählung hervorbringen wird.

– (vi)

[ *Versuche eine geschlossene Formel zu finden.*

[ Wie oben schon erwähnt, war die Formel ein reines Zufallsprodukt, welches entstand, als Ergebnisse der ersten paar Abzählreihen miteinander verglichen wurden. Hierbei fiel auf, daß die Platznummer bei jeder  $2^n$  Zahl immer die "1" ist, und bei jeder weiteren sich die ergebende Sitznummer immer um 2 erhöhte, bis schließlich die nächste  $2^{n+1}$  Zahl auftrat. (Dies ist unter anderem sehr schön in einem Diagramm zu sehen, wie zum Beispiel im Punkt

"Joe's Problem - Plot". Es wurde aufgrund dieser Kenntnis die obige Formel (naja herleiten möchte ich nicht sagen) erstellt.

[ Im weiteren Bearbeiten der Aufgaben, besonders bei A6.2, wurde deutlich, daß sich nicht so simpel auf 3er, oder 4er Abzählreime übertragbar war. Über eine Korrektur die feinen Unstimmigkeiten, wurde erfolgreich hinweggesehen, um eine Bearbeitung der Aufgabe 6.2 zu ermöglichen.

– (vii)

[ *Wie kann man die Berechnung anhand der geschlossenen Formel für den Rechner optimal übersetzen?*

[ Was heißt denn hier optimal?!? Die läuft doch ohne Probleme (zumindest für die 2er-Regel). Außerdem ist sie mehr als 75fach so schnell wie die erste (zugegeben, noch sehr provisorische) Prozedur. Außerdem errechnet sie die Platznummer von Zahlen mit um die 408986 Stellen ( $2055^{123456}$ ) innerhalb von weniger als 30 Sekunden (was nicht viel mehr ist als die Zeit, die Maple zum Errechnen der Zahl selbst benötigt). (Zumindestens auf meiner alten Krücke von Rechner). So etwas kann man schon recht "optimal übersetzt" nennen. (Wer möchte, darf widersprechen)

## – A 6.2

– (i)

[ Ein Jahr später steht wieder eine Neuwahl bevor, aber diesmal soll jeder siebte ausscheiden. Joe hat aber seinen Einfluss geltend gemacht, so dass er diesmal nicht anfangen muss, obwohl er Obergeheimler ist.

[ *Welche Teile der letzten Aufgabe kannst du Übertragen?*

[ Die Aufgabe (ii), (iii) und (iv) können als Gerüst übernommen werden, müssen nur auf jeden beliebigen Abzählrhythmus verallgemeinert werden. Hierfür betrachtet man als erstes den Vorgang des "Runterzählens" auf nur ein Element. Wenn "y" die Zahl des Ausscheidens ist, so ist schon mal klar, daß wenn eine Reihe modulo  $y=0$  Elemente hat die nächste Reihe mit dem gleichen (modulo  $y = \text{Rest}$ ) Element anfängt wie die aktuell und genau  $1/y$  Elemente ausscheiden (siehe hierzu "Joe's Problem.xls" - "Table 2", da es doch einen gewissen Teil an Vorstellungskraft benötigen würde, sollte man sich die folgenden Schritte lediglich bildlich vorstellen wollen).

Aber es soll ja auch Reihen geben, deren Anzahl von Elementen (Personen) nicht genau durch DIE Zahl (wird ab nun meist durch die Variable  $y$  wiedergegeben. Wurde sie zwar vorher auch schon ... aber ab jetzt halt offiziell) teilbar sind - zu den kommen wir jetzt:

Der nächste Fall wäre logischer Weise, daß die Elemente der Reihe nicht genau durch  $y$  teilbar sind, sondern ein Rest von 1 (auch als "modulo  $y = 1$ " ausgezeichnet) übrig

bleibt. In diesem Fall würden natürlich nicht  $1/y$  der Elemente eliminiert, sondern abhängig von dem Rest der ersten Zahl in der Reihe entsprechen, ist ein gewisser Summand in den Zähler mit einzuberechnen (es empfiehlt sich hier, die obige Graphik in Tabelle 2 anzuschauen ... für jede Anzahl von Elementen, wurden hier alle Möglichkeiten (Rest der ersten Zahl bezüglich modulo  $y$ ) aufgelistet). Das doch sehr simple Muster, hinter den ganzen Überlegungen wird sehr schön deutlich in der für die 6er-Regel angefertigten Tabelle (bunnt, bunnt). [Für den geistigen Notfall, einfach mal selber ne Zahlenreihe mit ca 20 Elementen aufschreiben, muß auch nicht unbedingt bei 1 anfangen. Jetzt je nach gewünschter Konstellation von beginnender Zahl und Anzahl der Elemente, mit 2 Gegenständen (Blätter eignen sich hier besonders gut) die entsprechenden unnützen Elemente davor und dahinter abdecken ...]. Das so gefundene Muster lässt sich dann in eine if-Abfrage umwandeln (siehe *Verbesserte allgemeine Prozedur* neben #1), wobei hier die von der Runde " $b$ " abhängige Variable " $k$ " (bzw. " $a$ ") das erste Element der entsprechenden Zeile und " $g$ " (bzw. " $z$ ") die Anzahl der Komponenten beschreibt. Eine nachfolgende Abfrage testet, ob nur noch ein Element übrig ist, für den positiven Fall beendet diese die Schleife und geht über zum 2. Teil des Programms - dem "Raufzählen".

Dieser Teil, der im ersten Aufgabenteil noch leicht abgesteckt werden konnte, erforderte doch schon ein wenig mehr Arbeit. Das Prinzip ist ähnlich - man versucht anhand der aktuellen Position und dem Charakter (Rest) der ersten Zahl der vorherigen (aus Sicht des Raufzählens natürlich der nächsten) Runden die neue Positionsnummer : ermitteln. Für die folgenden Erklärungen empfiehlt es sich wieder die erstellten Graphiken (diesmal sei ein besonderes Augenmerk auf die 3. Graphik der 2. Tabelle gerichtet) der Excel-Datei zu konsolidieren:

Es empfiehlt sich wiederum eine Reihe zu notieren, diesmal sollte man aber Variablen (Platzhalter oder auch Buchstaben - da es sonst zu einem großen Durcheinander kommen würde, da man ja zusätzlich Zahlen zwischen die schon notierten ergänzen muß aufschreiben. Nun überlegt man sich an welchen Stellen die (damals gestrichenen) Zahlen wieder eingefügt werden müssen (die verschiedenen Beispiele sollten dies vermitteln können). Als erstes sind mehrere Beispiele für den Fall, daß die erste Zahl d vorherigen Reihe mit dem Rest 1 anfing beschrieben. Man sieht hier sehr schön, daß in den Fall, daß vor der aktuellen Position 2 Elemente stehen es eine Positionsänderung von 1, und bei 4 davor stehenden eine Änderung von 2 usw. auftritt. Zusätzlich, ist aber auch zu beobachten, daß bei 3 davor stehenden Elementen die gleiche Änderung wie bei 2en eintritt. Es zählen also nur vollständige  $(y-1)$  (bei der 5er-Regel, gibt es also pro ganze 4 davorstehenden Elementen eine Positionsänderung) Elemente. Dies ist der Grund warum in der "verbesserten allgemeinen Prozedur" der Befehl "*trunc*" auftaucht, er ist für das Runden auf ganze Zahlen zuständig. Beginnen die Zeilen nicht mit dem Rest 1, muß jetzt nur noch die Überlegung angestellt werden, ab wievielen Elementen es zu einer Positionsänderung kommt, aufgrund der wieder eingefügten (beim runterzählen herausgestrichenen) Elemente, wobei bedacht werden muss, an welcher Stelle sie wieder eingefügt werden. Die angeführten Beispiele dürften dies vor Augen führen. Mit einigen weiteren logischen Überlegungen kann man nun auf die links neben den Beispielen stehenden Formeln (für die 3er-Regel) gelangen, und sie für die Allgemeinheit erweitern. Die allgemeine Form, wurde dann auch in eine if-Abfrage umgearbeitet, und ist unter #2 einzusehen.

## — Einige Worte noch zum Schluß ...

1. eine ordentliche Rechtschreibung war nicht unser höchster Anspruch, man soll es nur lesen können
2. ist nun ein bisschen mehr geworden als erwartet, zumindest der schriftliche Teil ...insbesondere die Erklärungen zur Verallgemeinerung unserer Prozedur. Tut uns echt leid,

ging aber nicht besser ...

3. Es wird im allgemeinen gehofft, daß die (sparsam gesäten) bunten Bildchen - wenn scho nicht zum Verständnis - wenigstens doch zur Unterhaltung beitragen konnten.

4. Die (lustigen)Abzählreime haben wir übrigens aus dem Netz also, wer noch Nachschub will (<http://www.zzebra.de/index.asp?themaId=618>) :-)

5. das wars dann woll auch, was noch gesagt werden mußte..

6. Ach ja ...

Ehne mehne Miste ....

( **not to be continued ...** )