

```
> restart;
' mod ':=modp:
```

## **Projekt 6: Joes Problem**

Verena Knop & Arnd Krömeke  
22.01.2003

### **Aufgabe P6.1**

#### **Was genau passiert in einer Runde?**

```
> runde:=proc(n)
  local Geheimler,i,p:
  Geheimler:=array(1..n):
  for i to n do
    Geheimler[i]:=i
  end do:
  p:=0;
  for i from 1 to n do

      if Geheimler[i]<>0 then p:=p+1:
      end if:

      if p=2 then Geheimler[i]:=0:
      p:=0:
      end if:

  end do:
  print (Geheimler);
end proc:
> runde(20);

      [1, 0, 3, 0, 5, 0, 7, 0, 9, 0, 11, 0, 13, 0, 15, 0, 17, 0, 19, 0]
```

#### **Einschub: Erklärung eines "Arrays"**

```
> arr:=proc(n)
  local i,Geheimler:
  Geheimler:=array(1..n):
  for i to n do
    Geheimler[i]:=i
  end do;
  print (Geheimler);
end proc:
> arr(20);

      [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

In einer Runde zählt der Computer also von 1 bis n alle Zahlen durch. Trifft er dabei auf die Zahl  $p=2$ , setzt er diese Zahl gleich null und zählt mit eins zählt mit eins wieder weiter. Somit wird also in einer Runde jede zweite Zahl gleich null gesetzt.

#### **Übertragung des Modells in den Computer:**

```

> joe:=proc(n)
  local Geheimler,i,p,q,r,s,t:
  Geheimler:=array(1..n):
  for i to n do
    Geheimler[i]:=i
  end do:

  p:=0:
  r:=n:
  t:=1:

  if n < 1 then print ("Bitte wähle eine Zahl größer 1!"):
  else:

    for q from 1 to n while r>1 do

      for i from 1 to n do

        if Geheimler[i]<>0 then p:=p+1:
        end if:

        if p=2 then Geheimler[i]:=0:
        p:=0:
        r:=r-1:
        end if:

      end do:

    end do:

  end if:

  for s from 1 to n do

    if Geheimler[s]<>0 then t:=Geheimler[s]:
    end if:

  end do:

  print("Die Person an der Stelle" ,t, "ist der neue
  Obergeheimler!");

end proc:
> joe(40);
      "Die Person an der Stelle", 17, "ist der neue Obergeheimler!"
> st:= time():
joe(20000);
time() -st;
      "Die Person an der Stelle", 7233, "ist der neue Obergeheimler!"
      2.954

```

Man sieht, dass dieses Programm auch bei großen Mengen die Platznummer des neuen Obergeheimlers langsam berechnen kann.

### Erstellen einer Tabelle:

```
> tab:=proc(n)
  local i,o:
  for i from 1 to n do
  print ([i]=joe2(i)):
  end do:
  end proc:
> tab(33);
```

```
[1]=1
[2]=1
[3]=3
[4]=1
[5]=3
[6]=5
[7]=7
[8]=1
[9]=3
[10]=5
[11]=7
[12]=9
[13]=11
[14]=13
[15]=15
[16]=1
[17]=3
[18]=5
[19]=7
[20]=9
[21]=11
[22]=13
[23]=15
[24]=17
[25]=19
[26]=21
[27]=23
[28]=25
[29]=27
[30]=29
[31]=31
```

$$[32]=1$$

$$[33]=3$$

An der Tabelle erkennt man, dass, wenn die Anfangsmenge eine Zweierpotenz ist, ist immer die Platznummer eins der neue Obergeheimler. Und ab diesen Zweierpotenzen folgen dann in aufsteigender Reihenfolge immer wieder ungerade Gewinner-Platznummern.

Dies gab uns Anlaß zu der Vermutung, dass die Platznummer des neuen Obergeheimlers abhängig ist von dem Zweierlogarithmus der Anfangsmenge.

Betrachtet man die Tabelle noch genauer, fällt auf, dass die neue Platznummer immer zweimal die Differenz der Anfangsmenge und der nächst kleineren Zweierpotenz plus eins ist. Beispiel: Anfangsmenge = 25, nächst kleinere Zweierpotenz ist  $2^4=16$ .

$$2*(25-16)+1=2*9+1=19$$

Ein Versuch zu unserer Vermutung:

```
> log[2](19);  
  
ln(19)  
ln(2)  
  
> evalf(%);  
4.247927513  
  
> floor(%);  
4  
  
> 2^4;  
16  
  
> 19-2^4;  
3  
  
> 2*(19 mod 2^4)+1;  
7
```

**Unsere Lösung:**

```
> joe2:=proc(n)  
log[2](n):  
floor(evalf(%)):  
2*(n mod 2^%)+1:  
end proc;  
  
> st := time():  
joe2(3^200000):  
time() - st;  
0.651  
  
> joe2(19);  
7
```

## Aufgabe P6.2:

Jeder Siebte fliegt raus und man braucht nicht mehr bei eins anfangen zu zählen.

```
> joe7:=proc(n,m) # n:= Anfangsmenge, m:= variable Anfangsposition
  local Geheimler,i,p,q,r,s,t,u:
  Geheimler:=array(1..n):
  for i to n do
    Geheimler[i]:=i
  end do:

  p:=0:
  r:=n:
  t:=1:

  if n < 1 then print ("Bitte wähle eine Zahl größer 1!"):
  else:

    for q from 1 to n while r>1 do

      for i from 1 to n do

        if Geheimler[i]<>0 then p:=p+1:
        end if:

        if p=7 then Geheimler[i]:=0:
        p:=0:
        r:=r-1:
        end if:

      end do:

    end do:

  end if:

  for s from 1 to n do

    if Geheimler[s]<>0 then t:=Geheimler[s]:
    end if:

  end do:
  u:=(t-1+m) mod n:#t gewinner bei anfangsposition 1, m ist
  variable anfangsposition, n ist die
  Gesamtpersonenzahl
  if u=0 then u:=n end if;
  print("Die Person an der Stelle" ,u, "ist der neue
  Obergeheimler!");

  end proc:
> joe7(10,5);
      "Die Person an der Stelle", 3, "ist der neue Obergeheimler!"
```

Wir konnten also die Eingabe des Modells mit Hilfe des "Arrays" anwenden auf den Fall, dass jede siebte Person herausfliegt und man nicht bei dem alten Obergeheimler, der Position eins, anfangen muß zu zählen.

### Verallgemeinerung:

```
> joeA:=proc(n,m,v) # n:= Anfangsmenge, m:= variable
Anfangsposition, v:=
Zählvariable
local Geheimler,i,p,q,r,s,t,u:
Geheimler:=array(1..n):
for i to n do
  Geheimler[i]:=i
end do:

p:=0:
r:=n:
t:=1:

if n < 1 then print ("Bitte wähle eine Zahl größer 1!"):
else:

  for q from 1 to n while r>1 do

    for i from 1 to n do

      if Geheimler[i]<>0 then p:=p+1:
end if:

      if p=v then Geheimler[i]:=0:
p:=0:
r:=r-1:
end if:

    end do:

  end do:

end if:

for s from 1 to n do

  if Geheimler[s]<>0 then t:=Geheimler[s]:
end if:

end do:
u:=(t-1+m) mod n: # t Gewinner bei Anfangsposition 1, m ist
variable
Anfangspositon, n:= Gesamtpersonenzahl
if u=0 then u:=n end if;
print("Die Person an der Stelle" ,u, "ist der neue
Obergeheimler!");
```

```
| end proc:  
[ > joeA(10,5,7);  
      "Die Person an der Stelle", 3, "ist der neue Obergeheimler!"  
[ > joeA(25,1,2);  
      "Die Person an der Stelle", 19, "ist der neue Obergeheimler!"  
[ > joeA(12,8,6);  
      "Die Person an der Stelle", 10, "ist der neue Obergeheimler!"  
[ >  
[ >  
[ >
```