Op 10 augustus 2024 kregen we van Albert en Willem een Programmeeruitdaging om een wachtwoordenmaker te maken. In Forth.
Dus, aan het werk:

versie 1: gewoon maar beginnen zonder plan

```
\ eerste poging:
: geldig? ( n -- n flag ) dup [char] ! [char] ~ 1+ within ;
```

Het lijkt wel handig om een woordje te hebben dat kijkt of een voorgesteld teken geldig is. Deze eenvoudige versie gebruikt WITHIN. Let op! WITHIN neemt het laatste teken (~) niet mee, dus +1 is nodig. Als het teken niet goed is wordt het teruggegeven zodat het als nieuwe bron voor de-willekeurig-teken-generator gebruikt kan worden. Later kunnen we extra testen zoals op O en 0 of I en l toevoegen...

```
\ : testgeldig-brute-force ( -- )
\ : geldig. ( n -- )  geldig? if emit else drop 42 emit then ;
\   [char] ~ 3 + 0 ?do i .  i geldig. ."  - " loop ;
```

En dit is de test waarmee ik zag dat ~ niet geldig werd verklaard. Het aantal mogelijkheden is voldoende klein om gewoon alles te laten zien...

```
: verander ( n -- n )
    begin 144 + 999 * 500 / 126 and geldig?
    if exit then again ;
```

Ik had geen zin op dit moment om de aloude formule op te zoeken, dus iets willekeurigs berekend op basis van de invoer. De 126 AND is een niet zo slimme denkfout, in plaats van 127 had ik de waarde van ~ genomen. Nu hebben we dus alleen even getallen omdat het laatste bitje 0 is. Jammer, gemiste kans. En het aantal mogelijke passwords is gehalveerd.
Nog iets wat mis kan gaan: als er alleen maar ongeldige tekens in de reeks zitten dan blijven we hangen in de begin/again lus. Dit had ik ook moeten testen, maar niet aan gedacht.

```
: 1ww1. ( n -- n )
    dup geldig? if dup emit else verander dup emit then ;
: ww1. ( x.n len.n -- )    0 ?do i + 1ww1. loop ;
```

De eigenlijke generator. Len gaat meteen naar de ?do loop; x wordt de hele tijd doorgegeven, verhoogd met **i +**
1WW1. Kijkt of de x geldig is, zo niet dan wordt x VERANDERd.
VERANDER levert altijd een geldige waarde, dus die kan altijd worden afgedrukt. Resultaat: 23 19 ww1. LMORV[ahpy$/;HVeu*< ok

versie 2: leren van je fouten

```
create willekeur 0 ,
```

Variable gebruik ik liever niet, dit geeft duidelijk aan wat er gebeurt. 1 cel met het adres van woord WILLEKEUR. En het is fexibeler ... voor als je meer cellen nodig hebt

```
: nw ( -- n )  willekeur @  31421 *  6927 +  dup willekeur ! ;
```

De aloude formule: elke NW genereert een nieuwe WILLEKEUR waarde

```
: 1ww2. ( -- )  nw 94 mod 33 + emit ;
\ laagste teken: 33. hoogste teken 126 = 93 + 33.
\ willekeurig getal 94 mod is een getal 0...93
```

Het is altijd goed om even te controleren. En om je test paraat te hebben als je een nieuw forth dialect gaat proberen:

```
\ : t 256 0 ?do 3 spaces i . i 94 mod 33 + dup . emit loop ;
: ww2. ( n.x n.len -- )  swap willekeur ! 0 ?do 1ww2. loop ;
```

Een versie met spaties, voor de leesbaarheid

```
: ww3. ( n.x n.len -- )
    swap willekeur ! 1+ 1 ?do
      1ww2. i 5 mod 0= if bl emit then
    loop ;
```

Resultaten:

```
23 19 ww2. mv=D{4GJ9j'b-072u<{ ok
```

En:

```
23 19 ww3. mv=D{ 4GJ9j 'b-07 2u<{ ok
```

Herhaalbaar, zoals gevraagd, en met spaties als rustpunten voor het oog

versie 3: ... en nu iets heel anders!

```
\ (origineel van Daniel Shawcross Wilkerson, 26 January 2009)
\ proquints (gebaseerd op Engels: pronouncable quintuples):
\ een getal wordt omgezet in een opeenvolging van medeklinkers
\ en klinkers om een uitspreekbaar woord te krijgen
\
\ een opeenvolging van medeklinkers m en klinkers k kunnen we
\ zo weergeven:
\ mkmkm = medeklinker, klinker, medeklinker etc. en
\ het woord dat hierdoor ontstaat is altijd uitspreekbaar.
\
\   Engelse medeklinkers: b d f g h j k l m n p s t v w z sh ch th
\   klinkers: a e i o u r
\ aanpassing voor Nederlandse uitspraak:
\   medeklinkers: b d f g h j k l m n p s t v w z sj sch pr
\   klinkers: a e i o u ij ui oe eu
\
\ nadeel van aanpassing aan nederlandse uitspraak is dat
\ de lengte in tekens niet constant is, dus:
\ we gebruiken b-z (16 stuks) voor 4 bits
\ en a e i u (4 stuks) voor 2 bits
\ dus een medeklinker m is 4 bits en een klinker k is 2 bits
\ en met de combinatie mkmkm (5 tekens) krijgen we
\ 3m + 2k = 3*4 + 2*2 = 16 bits en een
\ 5-letterig uitspreekbaar woord
\ uitwerking:
: u>c ( u ca x -- c ) drop + c@ ; \ hulpje
: 2bits ( u -- c ) 3 and s" eiau" u>c ; \ klinkers (k)
: 4bits ( u -- c ) 15 and s" bdfghjklmnpstvwz" u>c ; \ (m)
: m. ( u -- u' ) dup 15 and 4bits emit 4 rshift ;
: k. ( u -- u' ) dup 3 and 2bits emit 2 rshift ;
: eenklank.  ( u -- u' ) m. k. m. ; ( 10 bits)
: tweeklank. ( u -- u' ) m. k. m. k. m. ; ( 16 bits)
: drieklank. ( u -- u' ) m. k. m. k. m. k. m. ; ( 22 bits)
: vierklank. ( u -- u' ) m. k. m. k. m. k. m. k. m. ; ( 28 bits)
: vijfklank. ( u -- u' ) m. k. m. k. m. k. m. k. m. k. m. ; ( 24
bits)

cr ." ww4." cr
: ww4. (  x.n  -- x'.n )
    willekeur !
    nw dup tweeklank. bl emit tweeklank. bl emit drop ;

46238684326 ww4. cr ww4. cr ww4. cr  ww4. cr  ww4. cr  ww4. cr
ww4. cr drop
```

```
cr ." ww5." cr
: ww5. (  n.x  -- n.x' )
    willekeur !
    nw dup vijfklank. bl emit tweeklank. bl emit drop ;

2376474628323764 ww5. cr ww5. cr ww5. cr  ww5. cr  ww5. cr  ww5.
cr  ww5. cr drop
```

Resultaat:

```
ww4.
vikis luzip
muleb wanub
latus labid
pigev bunev
debeg figef
telib wemik
sapan vitib

ww5.
gugugiliwas wevaj
kudawimunan kepuw
vamizehahil vusug
meduhujihet fusah
lubusiguwez tuhib
pakihajanuk gamaw
dikumenedaw hesus
```

-- FIN--