Musteraufgabe

Zwei und vierzig

Banken verlangen häufig, daß Zahlen nicht nur in Ziffern, sondern auch in Worten angegeben werden.

Beispiel:	42	zweiundvierzig
	425	vierhundertfünfundzwanzig
	1300	eintausenddreihundert
	24201	vierundzwanzigtausendzweihunderteins
	101000	einhunderteintausend
	400016	vierhunderttausendsechzehn

Aufgabe:
Schreibe ein Programm, welches natürliche Zahlen zwischen 1 und 999999 korrekt in Worten ausgeben kann.
Erläutere ausführlich das Konstruktionsprinzip.

Lösungsidee und Programmdokumentation:

Prinzipiell werden deutsche Zahlwörter vergleichsweise regelmäßig gebildet:
Aus 735629 wird 7 **hundert** 5 **und** 30 **tausend** 6 **hundert** 9 **und** 20.
Ausnahmen bilden dabei **elf, zwölf, sechzehn** (weil das **s** am Ende der 6 fehlt) und **siebzehn** (es heißt nicht **siebenzehn**).
Auch bei den Zehnern gibt es diverse Ausnahmen, nämlich **zwanzig, dreißig, sechzig** und **siebzig.**
Am einfachsten geht man mit diesen Ausnahmen um, indem man die Zahlen von 1 bis 20 und alle Zehner als Konstanten vereinbart.
Schließlich muß man noch auf die 1 aufpassen, denn bei auf 01 endenden Zahlen muß es **eins** heißen, innerhalb der Zahlwörter jedoch **ein**.

Für die Konstanten von 1 bis 19 verwenden wir ein Array *klein*, für die Zehner ein Array *zig*. Dann definieren wir drei Funktionen *zehner, hunderter* und *tausender*, die für die korrekte Umwandlung der Zahlen von 0 bis 99, von 100 bis 999 und von Zahlen ab 1000 sorgen.
Am einfachsten ist dabei die Funktion *tausender,* da sie nur wieder die Funktion *hunderter* aufruft und das Wort **tausend** einfügt.
In der Funktion *hunderter* wird mittels Division durch 100 berechnet, wieviele Hunderter in der Zahl enthalten sind und dafür gesorgt, daß nur dann das Wort **hundert** eingefügt wird, wenn es tatsächlich gebraucht wird. Der noch nicht in Buchstaben verwandelte Teil der eingegebenen Zahl wird mittels der Modulo-Funktion ermittelt und wiederum an die Funktion *hunderter* übergeben.
Wenn es im Rest keine Hunderter mehr gibt, wird die Funktion *zehner* aufgerufen.

In der Funktion *zehner* wird überprüft, ob für die umzusetzende Zahl eine Konstante vereinbart wurde. Wenn ja, wird diese eingesetzt, ansonsten wird die Zahl in ihren Zehner- und ihren Einerteil aufgespaltet (wieder mittels DIV und MOD) und die Wörter für diese beiden Teile werden, falls nötig, durch **und** verbunden.

Die Berechnung des Wortes für die eingegebene Zahl wird gestartet durch den Aufruf der Funktionen *tausender(zahl* DIV 1000) und *hunderter(zahl* MOD 1000). Die Ergebnisse der Funktionsaufrufe werden aneinandergehängt.
Schließlich wird vor der Ausgabe noch geprüft, ob das ermittelte Wort auf **ein** endet, und wenn ja, wird ein **s** angehängt.

Die Eingabe von zu großen Zahlen führt nur zur Ausgabe **„zu groß!!! “**, danach können weitere Zahlen eingegeben werden. Beendet wird das Programm mit der Eingabe der Null. Die Eingabe von Zeichen statt Ziffern wird nicht abgefangen, sie führt zum Absturz des Programms.

Halbformale Programmbeschreibung:

Das Programm ist so kurz, daß jede halbformale Programmbeschreibung, die mehr enthält als nur

Einlesen der Zahl
Umwandeln in Zeichen
Ausgeben

nur unwesentlich kürzer wäre als das Programm. Außerdem sollte die Vorgehensweise nach der ausführlichen Erläuterung der Lösungsidee anhand des Programmtextes leicht nachvollziehbar sein.

Bildschirmausgabe:

```
0 <= Zahl <= 999999: 1001
eintausendeins
0 <= Zahl <= 999999: 666
sechshundertsechsundsechzig
0 <= Zahl <= 999999: 777
siebenhundertsiebenundsiebzig
0 <= Zahl <= 999999: 999999
neunhundertneunundneunzigtausendneunhundertneunundneunzig
0 <= Zahl <= 999999: 654321
sechshundertvierundfünfzigtausenddreihunderteinundzwanzig
0 <= Zahl <= 999999: 3432
dreitausendvierhundertzweiunddreißig
0 <= Zahl <= 999999: 75850
fünfundsiebzigtausendachthundertfünfzig
0 <= Zahl <= 999999: 1000000
zu groß!!!
0 <= Zahl <= 999999: 0
```

```
program zahlwoerter;   (* Lothar Oppor *)

var
  klein : array[1..19] of string[10];
  zig   : array[2..9] of string[10];
  zahl  : longint;
  ergebnis: string;

procedure init;
begin
  klein[1] := 'ein';
  klein[2] := 'zwei';
  klein[3] := 'drei';
  klein[4] := 'vier';
  klein[5] := 'fünf';
  klein[6] := 'sechs';
  klein[7] := 'sieben';
  klein[8] := 'acht';
  klein[9] := 'neun';
  klein[10] := 'zehn';
  klein[11] := 'elf';
  klein[12] := 'zwölf';
  klein[13] := 'dreizehn';
  klein[14] := 'vierzehn';
  klein[15] := 'fünfzehn';
  klein[16] := 'sechzehn';
  klein[17] := 'siebzehn';
  klein[18] := 'achtzehn';
  klein[19] := 'neunzehn';
  zig[2] := 'zwanzig';
  zig[3] := 'dreißig';
  zig[4] := 'vierzig';
  zig[5] := 'fünfzig';
  zig[6] := 'sechzig';
  zig[7] := 'siebzig';
  zig[8] := 'achtzig';
  zig[9] := 'neunzig';
end; (* init*)


function zehner(z : longint) : string;

begin
  if z < 1 then zehner := ''
  else if z <= 19 then zehner := klein[z]
  else if (z mod 10) = 0 then
    zehner := zig[z div 10]
  else zehner := klein[z mod 10] + 'und'
            + zig[z div 10]
end; (* zehner *)


function hunderter(z: longint) : string;

begin
  if z < 1 then hunderter := ''
  else if z < 100 then hunderter := zehner(z)
  else hunderter := klein[z div 100] +
            'hundert' + hunderter(z mod 100)
end; (* hunderter *)


function tausender (z : longint) : string;

begin
  if z < 1 then tausender := ''
  else tausender := hunderter(z) + 'tausend'
end; (* tausender *)

begin
  init;
  zahl := 0;
  while zahl >= 0 do
    begin
      write('0 <= Zahl <= 999999: ');
      readln(zahl);
      if zahl <= 0 then exit
      else if zahl <= 999999 then
      begin
        ergebnis := tausender(zahl div 1000) +
              hunderter(zahl mod 1000);
        if copy(ergebnis,length(ergebnis)-2,3) =
              'ein' then
          ergebnis := ergebnis + 's';
        writeln(ergebnis)
      end
      else
        writeln('zu groß!!!')
    end  (* while *)
end.
```

Design: Kuhn, Kammann & Kuhn, Köln Illustration: Gunnar Colfen, Christina Feilhauer, Köln Druck: DFS Druck und Verlag, Köln