

OPL Workshop 9 - Mein Programm lernt Englisch

Ulrich Krinzner, 01/2000

Kommerzielle Programme verwenden eine bestimmte Technik, um sich jeweils in landesgerechter Sprache dem Nutzer zur Verfügung zu stellen. Sie setzen dabei sogenannte Ressourcen-Dateien ein, die verschiedene Textversionen enthalten. Wir zeigen die Möglichkeit, eine vergleichbare Technik auch für kleinere OPL-Programme zu benutzen.

Wir haben uns als Beispiel ein jahreszeitgemäßes Programm ausgesucht: Wenn bei winterlichen Minusgraden zusätzlich noch ein rauhes Lüftchen weht, empfindet der Mensch eine weit tiefere Temperatur. Damit wir die richtige Kleidung aus dem Schrank fischen, haben Wissenschaftler eine Formel bestimmt, die uns diese "gefühlte" Temperatur (die sogenannte Windchill Temperatur) aus gemessener Temperatur und Windstärke errechnen läßt.

Das Problem

Weil wir (und Sie) nicht selbstlos sind, haben wir (und Sie) die Absicht, unser großartiges Programm auch einigen englischsprachigen Freunden - natürlich in deren Sprache - zur Verfügung zu stellen.

Würde es sich dabei um ein umfangreiches Programm handeln, müßte man also zwei verschiedensprachige Versionen aufbereiten. Das ist aber ein Unterfangen, das oft genug mit dem Auftreten zusätzlicher Fehler bestraft wird. Sicherheitshalber wären dann also in der Praxis zwei Programme zu testen - ein hoher Aufwand!

Eine Lösung

Wir haben eine bessere Lösung und die sieht wie folgt aus (Die zugehörigen Quelltexte finden Sie hinten am Ende des Workshops):

Wir spalten unser Programm in zwei Teile auf. In dem einen, dem Hauptprogramm, steht der "aktive" Teil des Programmes mit Schleifen, Bildschirmausgabe, Berechnungen und so weiter. Der andere Teil enthält nur den Text, entweder in Deutsch oder in Englisch. Die OPL-Quelltexte müssen wie üblich nach dem Erstellen durch "Übersetzen" lauffähig gemacht werden, bevor sie in dieser Form zusammen an einen Nutzer weitergegeben werden können. Die übersetzten Programme nennt auch Module - dieser Begriff wird immer wieder auftauchen.

So geht's praktisch

Statt der unmittelbaren Texte, die wir im Verlaufe des Programmes ausgeben wollen, verwenden wir im Hauptprogramm nun Textvariablen (Strings). Je nach verwendeter Sprache weisen wir diesen Textvariablen über ein Unterprogramm die jeweils anderssprachige Texte zu. Das Unterprogramm liegt einschließlich aller Texte , wie gerade zuvor erläutert, in eben jenem Textmodul.

Jede Sprache bekommt ihre eigene OPL-Datei, damit man sie bereits äußerlich erkennen kann. Bei uns heißen sie "text_d" und "text_e". Achtung! Nach dem Übersetzen muss die jeweils zum Einsatz kommende OPO-Datei noch in "text.opo" umbenannt werden, sonst geht's nicht!

Auf die beschriebene Art wäre es ziemlich einfach, auch noch eine spanische oder eine andere Sprachvariante zur Verfügung zu stellen, wenn Sie diese Sprache nur hinreichend beherrschen.

"Lade mich"

Damit das alles aber auch zusammen funktioniert, muss dem Hauptprogramm der Inhalt des Textmoduls zur Verfügung stehen. Aus vorhergehenden Workshops wissen Sie, dass eine Datei ihren Inhalt nur preisgibt, wenn man sie zu diesem Zwecke vorher "öffnet". In unserem Falle ist das ein spezielles Öffnen - ein "Laden". Der Lade-Befehl lautet allgemein: LOADM("modulname.opo").

Innerhalb der Startroutine "PROC main:" des Hauptprogramms wird nach dem Laden des Textmoduls auf das darin befindliche Unterprogramm "PROC Satz\$(i%)" zugegriffen, um eine Textvariable nach der anderen mit Text zu füllen. Die dabei übergebene Integer-Zahl ist ein Zeiger ("Vektor" im

Sprachgebrauch der Programmierer) auf eine Markierung im Unterprogramm, an der der benötigte Wert ohne lange Sucherei abgeholt werden kann.

Ein Unterprogramm denkt

Werfen Sie einen Blick auf die unten abgedruckte "PROC Satz\$(i%)". In unserer Sprache ausgedrückt, reagiert das Programm etwa so:

"Ich habe den Zeiger "2" über den Parameter i% übergeben bekommen, also soll ich zur zweiten Marke gehen. Der Name der zweiten Marke lautet "mm2", denn das ist so in der Liste zwischen VECTOR i% und ENDV festgelegt. Also springe ich nun zur Marke "mm2:". Dort hole ich den entsprechenden Text ab und gebe ihn sofort an das aufrufende Programm weiter."

Das gleiche Unterprogramm ließe sich auch mit einer IF/ELSEIF .. /ENDIF Konstruktion lösen, nur wäre es dann durch die immer wieder neu zu treffenden Entscheidungen deutlich länger beschäftigt.

Speichergrenze

Sowie alle Textvariablen ihren Text haben, wird das Modul durch einen UNLOADM-Befehl wieder "entladen". Damit wird auch der Speicher, der durch das Laden belegt wurde, wieder freigegeben. Ein weiterer Grund, überhaupt mit Modulen zu arbeiten. Das war/ist besonders für die Rechner der Serie 3xx wichtig, da dort eine 64kB-Grenze besteht. Programme, die mehr als 64kB benötigen, funktionieren nicht. Also teilt man sie in mehrere Module auf und entfernt jeweils nicht benötigte aus dem Speicher. Wenn Sie sich von der Grenze überzeugen wollen: Versuchen Sie einmal mit WORD einen Text zu bearbeiten, der größer als 40kB ist ...

Wandel der Werte

Zur Übung haben wir das Vektor-Verfahren noch einmal in "PROC wind:(i&)" verwendet. Darin werden die Windstärken in die Windgeschwindigkeit (in km/h) umgewandelt, die für die Formel gebraucht wird. Das ginge über ein Variablenfeld etwas einfacher, aber als Übung ist es recht brauchbar.

Achten Sie auf die Bezeichnung des Unterprogrammes. Im ersten Fall hatten wir es mit der Rückgabe eines Textes zu tun - der Name des Unterprogrammes war deshalb gleich mit dem Anhängsel "\$" versehen, ganz wie auch bei Stringvariablen üblich. Andernfalls funktioniert es nicht! Im zweiten Falle handelt es sich um die Rückgabe einer floating point Zahl, eine besondere Kennzeichnung des Unterprogramm-Namens entfällt daher.

Es gibt in "PROC wind:(i&)" zwei Besonderheiten:

- 1) Da die Windstärken in unserem Programm erst bei "2" anfangen, wird "i&" durch Subtraktion einer "1" einfach noch einmal angepaßt, damit die Listenkonfiguration wieder stimmt.
- 2) Der Übernahmeparameter "i&" ist eine Longinteger-Zahl! Das resultiert aus dem Umstand, dass der zugehörige Dialog (dLONG v&,vtext\$,2,9) nur für den Einsatz von Longinteger-Zahlen ausgelegt ist und mit diesem Wert gleich weitergearbeitet wird (spart eine zusätzliche Variable).

Dialogbereitschaft

Über Dialoge haben Sie bereits in unserer Ausgabe 11 etwas erfahren. Damals haben wir allerdings nicht alle Möglichkeiten ausgeschöpft. Heute sind zwei weitere hinzugekommen.

Zuerst ist da die Eingabemöglichkeit für Integerzahlen, oder (wie gerade erwähnt) genauer: Longinteger-Zahlen, der ja den Bereich der "normalen" Integerzahlen mit beinhaltet. Zur Erinnerung: Longintegerzahlen (Variablenkennung"&") umfassen alle ganzzahligen Werte im Bereich -2.147.483.648 bis +2.147.483.647, einfache Integers (Kennung "%") dürfen ab -32.768 beginnen und bis +32.767 reichen.

Innerhalb des Dialogbereiches fügt man dazu eine Zeile mit folgendem Inhalt ein:

```
dLONG eingWert&,"Text",minWert&, maxWert&
```

Nach dem Dialog steht der gewählte bzw. eingetragene Wert in der Variablen "eingWert&". Diese kann nur Werte im Bereich zwischen den Grenzen minWert& und maxWert& (jeweils einschließlich) annehmen. Die Grenzen gibt man selber vor.

Die nächste im Dialog anzugebende Zahl ist vom Typ "floating point", also eine Fließkommazahl. Daher benutzt man als zweite Eingabemöglichkeit diese Programmzeile:

```
dFLOAT eingWert,"Text", minWert&, maxWert&
```

Das gleicht der vorherigen Eingabemöglichkeit bis auf eine Ausnahme - natürlich - die Variable "eingWert" beherbergt eine Fließkommazahl.

Alle anderen Programmierweisen sind Ihnen bereits vertraut, wenn Sie den Workshop von Anfang an verfolgt haben.

Da sich das Programm immer darauf verläßt, dass ein Textmodule "text.opo" existiert, vergessen Sie vor dem Verteilen an Verwandte und Bekannte nicht, das jeweils zutreffende Text-Modul umzubenennen!

Wenn Sie experimentieren wollen: Bauen Sie einmal in das Hauptprogramm eine Umschaltmöglichkeit für die Sprache ein und organisieren Sie das Textmodul entsprechend um.

Also dann - ziehen Sie sich warm an und wählen Sie die passende Kleidung! Lassen Sie sich dabei von unserem kleinen Programm helfen ... egal, welche Sprache Sie sprechen!

Hauptprogramm, bestehend aus den Teilprogrammen "PROC main:", "PROC rechne\$(t,w&)" und "PROC wind:(i&)":

```
PROC main:
    LOCAL temp, v&                REM Temp.(°C) u. Windstaerke
    LOCAL titel$(64), ttext$(64), vtext$(64)
    LOCAL text1$(64), text2$(64), text3$(64), result$(16)

    LOADM "text.opo"
        titel$= satz$(1)
        ttext$= satz$(2)
        vtext$= satz$(3)
        text1$= satz$(4)
        text2$= satz$(5)
        text3$= satz$(6)
        result$= " "
    UNLOADM "text.opo"

    DO
        dINIT titel$
            dLONG v&,vtext$,2,9
            dFLOAT temp,ttext$,-25,0
            dTEXT text1$,result$, $200
            dBUTTONS text2$,27,text3$,13
        IF DIALOG
            result$= rechne$(temp,v&)
        ELSE
            STOP
        ENDIF
    UNTIL 0
ENDP
```

```

PROC rechne$(t,w&)
  LOCAL twc      REM Windchill-Temperatur
  LOCAL v        REM Windgeschwindigkeit in km/h
  LOCAL zw, i    REM Zwischenwerte
  v= wind:(w&)  REM Windstaerke in Windgeschw. wandeln

  REM die Windchill-Formel, mit:
  REM t= Temperatur in Grad C, w&=Windstaerke,
  REM SQR()= Wurzel aus ..
  twc= 33+(0.478+0.237*SQR(v)-0.0124*v)*(t-33)

  REM Rundung des Ergebnisses:
  zw= 10*twc
  i= INT(zw)
  IF ABS(zw-i)>0.5
    i=i-1
  ENDIF
  i=i/10
  REM Rundung beendet

  REM Ergebnis als String zurueckgeben:
  RETURN FIX$(i,1,6)
ENDP

PROC wind:(i&)
  VECTOR i&-1
    mm1, mm2, mm3, mm4, mm5
    mm6, mm7, mm8
  ENDV

  mm1:: RETURN 8.5
  mm2:: RETURN 15.5
  mm3:: RETURN 24.0
  mm4:: RETURN 33.5
  mm5:: RETURN 44.0
  mm6:: RETURN 55.5
  mm7:: RETURN 68.0
  mm8:: RETURN 81.5
ENDP

```

Modul text_e (übersetzt: text_e.opo)

```

PROC satz$(i%)      REM englische Version
  VECTOR i%
    mm1, mm2, mm3, mm4, mm5, mm6
  ENDV
  mm1:: RETURN "Calculator for Windchill Temperature"
  mm2:: RETURN "Temperature? (0 .. -25°C)"
  mm3:: RETURN "Wind strength? (2 .. 9)"
  mm4:: RETURN "Felt temperatur in °C is: "
  mm5:: RETURN "Quit program"
  mm6:: RETURN "Calculate"
ENDP

```

Modul text_d (übersetzt: text_d.opo)

```

PROC satz$(i%)      REM deutsche Version
  VECTOR i%
    mm1, mm2, mm3, mm4, mm5, mm6
  ENDV
  mm1:: RETURN "Windchillberechnung"
  mm2:: RETURN "Temperatur? (0 .. -25°C)"
  mm3:: RETURN "Windstärke? (2 .. 9)"
  mm4:: RETURN "Gefühlte Temperatur in °C ist: "
  mm5:: RETURN "Beenden"
  mm6:: RETURN "Berechnen"
ENDP

```

ENDP

Calculator for Windchill Temperature

Wind strength? (2 .. 9) ◀ 2 ▶

Temperature? (0 .. -25°C) 0

Felt temperatur in °C is:

To quit use <Esc> key ...

Windchillberechnung

Windstärke? (2 .. 9) ◀ 2 ▶

Temperatur? (0 .. -25°C) 0

Gefühlte Temperatur in °C ist:

<Esc> beendet ...