Portabler Maskengenerator für Microcomputer

Zusammenfassung: Die Anforderungen an die Benutzerfreundlichkeit von Bildschirmdialogen sind in den letzten Jahren gestiegen, resultierend aus der steigenden Zahl der ungeübten Benutzer. Diesem Nebeneffekt der verteilten Rechnerleistung kann nur durch den Einsatz von automatischen Programmierhilfsmitteln begegnet werden. Der nachfolgende Vortrag beschreibt die Realisierung der wichtigsten Ergonomieforderungen durch ein teilweise nichtprozedurales Softwarewerkzeug für Microcomputer.

1. Ergonomieforderungen an Bildschirmdialoge

Der Arbeitsplatzrechner bringt immer mehr ungeübte Benutzer mit dem Medium Bildschirm zusammen. Dadurch müssen erhöhte Anforderungen an die Interaktivität, Störsicherheit und kognitive Effizienz von Software gestellt werden. Da über diese Anforderungen jedoch verschiedene Meinungen bestehen, möchten wir hier die unserer Meinung nach unverzichtbaren Grundforderungen erläutern. Wir beziehen uns dabei auf die schon 1973 von Martin /2/ gestellten Forderungen:

Kontinuität:

Die Masken sollen in allen Programmen ähnlichen Aufbau haben, selbst wenn mehrere Programmierer an einem Projekt arbeiten. Der Benutzer bekommt so ein vertrautes Bild und wird sicherer im Umgang mit den Programmen /1,2/.

Übersichtlichkeit:

Die Masken sollen in einen Überschriftsteil, der die Bedeutung der momentanen Maske erläutert, in einen Informationsteil, in dem die Ein/Ausgaben erfolgen, und in einen Fußteil, in dem Fehlermeldungen und Bedienungshinweise ausgegeben werden, aufgeteilt sein. Untersuchungen weisen darauf hin, daß sich diese Darstellung als besonders vorteilhaft in Bezug auf das Benutzerverhalten erwiesen hat.

Zusätzliche Bedeutung hat die Verwendung von Bildschirmattributen wie doppelte Helligkeit, Blinken, Farbe etc., um wichtige Eingaben oder Fehlermeldungen besonders zu kennzeichnen. Hier ist jedoch mit Bedacht vorzugehen, da die kritiklose Verwendung von Attributen wieder zu Unübersichtlichkeit führt.

Neben diesen allgemeinen Ordnungsvorschriften für die Maske sollen die alphanumerischen Eingabefelder linksbündig, die numerischen rechtsbündig auf dem Schirm erscheinen. Jedes Eingabefeld soll mit einer aussagefähigen Bezeichnung versehen sein /2/.

Fehlertoleranz 1. Art:

Unter Fehler 1. Art sind die syntaktischen Fehler zu verstehen. Felder, in denen Zahlen eingelesen werden sollen, dürfen keine alphanumerischen Zeichen annehmen. Im Fehlerfall soll ein ungültiges Zeichen sofort verweigert werden und ein akustisches Signal ertönen. Durch weitere Formate (außer Numerisch / Alphanumerisch) lassen sich Eingabeklassen für Zeichen bilden, so daß viele Eingabefehler ohne Belastung des Benutzers verhindert werden.

Fehlertoleranz 2. Art:

Unter Fehler 2. Art sind die semantischen Fehler zu verstehen. Nach der Eingabe zum Beispiel einer Postleitzahl soll vom Dialogprogramm sofort der Bereich getestet werden (in Deutschland 1000-8999). Im Fehlerfall soll der Cursor wieder auf das Eingabefeld positioniert werden und in der Fußzeile ein Hinweis erscheinen. Der Benutzer kann so seine fehlerhafte Eingabe sofort korrigieren und muß nicht später noch einmal in seinem Arbeitsfluß stoppen.

Komfort:

Es muß in Eingabefeldern die Möglichkeit gegeben sein,

- vorgegebene Werte durch Drücken der Datenfreigabetaste zu übernehmen,
- solche Startwerte abzuändern,
- eingegebene Werte durch die Funktionen Löschen, Einfügen und Überschreiben zu ändern,
- Feldgrenzen sichtbar zu machen,
- und nach dem Verlassen eines Eingabefeldes dieses einfach wieder zu erreichen, um Korrekturen durchführen zu können.

Funktionstasten:

Funktionstasten bieten die Möglichkeit, in Menümasken eine schnelle Auswahl zu treffen und gewisse Hilfsfunktionen zur Verfügung zu stellen, wie Erläuterungsschirm, Eingabe abbrechen etc. Sie bieten, wenn sie einheitlich verwendet werden, große Vorteile.

2. Codieraufwand kontra Benutzerfreundlichkeit

Die obigen Forderungen sind zwar allgemein bekannt und zum Teil auch akzeptiert. Es finden sich aber nur selten Programme, die sie erfüllen. Dafür gibt es nach unserer Meinung folgende Gründe:

- Existierende Programmiersprachen, mit Ausnahme einiger Ansätze in Cobol, unterstützen den Programmierer nur ungenügend bei der Realisierung der Forderungen.

- Dem Entwurf einer Maske auf Papier folgt eine zeitraubende Umsetzung in Code. Die Positionen der Ein/Ausgaben müssen mühsam abgezählt werden. Zudem weist der Entwurf auf Papier den Nachteil auf, daß nicht mit dem Layout experimentiert werden kann, daß also Zeilen nur durch Neuzeichnung verschoben, eingefügt oder ausgerichtet werden können.

- Die Realisierung der Ergonomieanforderungen setzen eine detaillierte Kenntnis des Bildschirms voraus. Dadurch wird das Programm von einer zur Codierzeit gegebenen Hardwarekonfiguration abhängig.

- Existierende Hilfsmittel sind Sprach- oder Rechnerabhängig.

- Benutzer der Programme verschweigen ihre Bedürfnisse beziehungsweise kennen sie mangels Vergleichsmöglichkeiten nicht. Im Falle von Standardsoftware sind Benutzer und Programmierer einander unbekannt.

3. Konfliktlösung mit Maskengenerator LARVE

Um die oben genannten Ergonomieforderungen unter Beibehaltung konventioneller Programmiersprachen zu erfüllen, wählten wir den Ansatz der Spracherweiterung und entwickelten das Softwarewerkzeug LARVE. Dieses Werkzeug besteht aus einem Maskengenerator, einem Formatinterpreter, einem Terminalkonfigurator und diversen Hilfsprogrammen wie Editor, Lernprogramm etc.

Entwurf:

Der Entwurf der Bildschirmmaske, evtl. unter Einbeziehung des Benutzers, geschieht mit einem beliebigen, bildschirmorientierten Editor. Dabei wird die Maske auf dem Bildschirm genau so entworfen, wie sie später zur Laufzeit auf dem Schirm erscheinen soll.

Die Eingabefelder werden durch Formatspezifikationen dargestellt, die später dem Formatinterpreter dazu dienen, die Eingabezeichen auf Zugehörigkeit zu Eingabeklassen (syntaktische Fehler) zu prüfen. Diese Formatspezifikationen werden in Begrenzungszeichen eingeschlossen. Die Anzahl der Elemente einer Formatspezifikation spiegeln die Länge der Eingabe wider.

In dieser Entwurfsphase kann und soll der spätere Benutzer, soweit möglich, zugegen sein, so daß auf seine speziellen Bedürfnisse eingegangen werden kann. Dadurch, daß das spätere Programm 'seine' Masken präsentiert, wird ein erster Schritt in Richtung Akzeptanz getan.

Die Maske ist schon während der Entwurfsphase in ihrer endgültigen Form sichtbar, liegt also nicht mehr abstrakt vor. Dieses nicht prozedurale Vorgehen verkürzt die Entwicklungszeit erheblich und schafft Raum für kreative Maskengestaltung.

Nach dem Entwurf wird die Maske als Textdatei abgespeichert. Sie kann so direkt zu Dokumentations- und Diskussionszwecken ausgedruckt werden.

Übersetzung:

Die Textdatei wird vom Generator, der in drei Schritten abläuft, im ersten Schritt durch einen Parser in eine interne Darstellung überführt. Syntaktische Fehler der Maske werden größtenteils korrigiert und auf einer Fehlerdatei mitprotokolliert.

In einem zweiten Schritt kann interaktiv die spätere Abarbeitungsreihenfolge der Eingaben, die ja problemspezifisch ist, angegeben werden.

In einem dritten Schritt erzeugt der Generator Prozeduren in der jeweils verwendeten Programmiersprache (z.B C, PL/1, Pascal, Basic etc.). Diese Prozeduren enthalten die Aufrufe an den Formatinterpreter mit den Koordinaten auf dem Schirm, den Formatspezifikationen sowie Standardannahmen für die Startwerte. Eine gegebene Maske wird also in ein Skelettprogramm überführt, das dann vom Programmierer wie gewohnt um die Semantik des Programms ergänzt werden muß.
Die Generatoren für verschiedene Programmiersprachen unterscheiden sich nur in ihrem letzen Schritt, der Maskenentwurf ist vollkommen sprachunabhängig.

Durch die automatische Übersetzung ist das erzeugte Programmgerüst sehr übersichtlich und fehlerfrei. Die Übersetzung einer Maske in Programmcode geschieht innerhalb einer Minute.

Ergänzung:

Der Programmierer ergänzt wie schon erwähnt, das Programmgerüst um die prozedurale Ablauflogik und um die semantische Fehlerbehandlung. An dieser Stelle kann er die Standardstartwerte gemäß seinen Anforderungen ändern, Benutzerhinweise ausgeben etc.

Laufzeit:

Zur Laufzeit wird durch das nun ergänzte und compilierte Programm bei jeder Eingabe der hinzugebundene Formatinterpreter aufgerufen. Der Formatinterpreter führt die Eingabe anhand der übergebenen Parameter durch. Mit Hilfe der Formatspezifikation prüft er jedes von der Tastatur eingegebene Zeichen auf syntaktische Gültigkeit und verweigert es gegebenenfalls.

Er stellt in den Eingabefeldern die Funktionen Löschen, Einfügen, Überschreiben sowie Cursorbewegungen nach links und rechts zur Verfügung. Ein Eingabefeld kann nur durch Drücken der Datenfreigabetaste oder einer Funktionstaste verlassen werden, nicht jedoch durch andere Eingaben oder durch die Cursorbewegungen.

Zur Sichtbarmachung der Eingabegrenzen können bei alphanumerischen Feldern Unterstreicher, bei numerischen Feldern Punkte als Startwerte vorgegeben werden, falls keine andere sinnvolle Vorgabe möglich ist. Diese Zeichen werden nach Verlassen des Eingabefeldes vom Formatinterpreter automatisch ausgefiltert.

Die Eingabe wird dann an das rufende Programm übergeben und kann dort weiterverarbeitet werden. Zusätzlich notwendige semantische Prüfungen finden dann auf Hochsprachenebene statt.

Portabilität:

Die Lauffähigkeit der Programme auf verschiedenen Bildschirmen wird durch einen Terminalkonfigurator gewährleistet. Der Formatinterpreter bezieht seine Informationen über Bildschirmsteuerung und Tastenbelegung aus einer internen Tabelle, die mit dem Konfigurator entweder interaktiv oder über ein Menü aus vorgegebenen Terminals verändert werden kann.

Der Programmierer arbeitet in seinem Programm nur mit logischen Funktionstasten, logischen Bildschirmattributen und den eingefügten Funktionen. So wird es möglich, ein Programm im Feld an die Hardware anzupassen. Ob ein Benutzer des Programms Fehlermeldungen blinkend oder in doppelter Helligkeit angezeigt bekommt, oder die Tastenbelegung entsprechend seinen Wünschen verändert werden soll, bedeutet nur einen Lauf des Konfigurators, nicht mehr ein Neucompilieren.

Die Zahl der Bildschirmattribute ist auf vier begrenzt, die Zahl der physischen Tasten für eine logische Taste auf drei. So können moderne Terminals gut ausgenutzt werden, ohne Hardwareabhängigkeit zu provozieren.

4. Anwendungsbeispiel

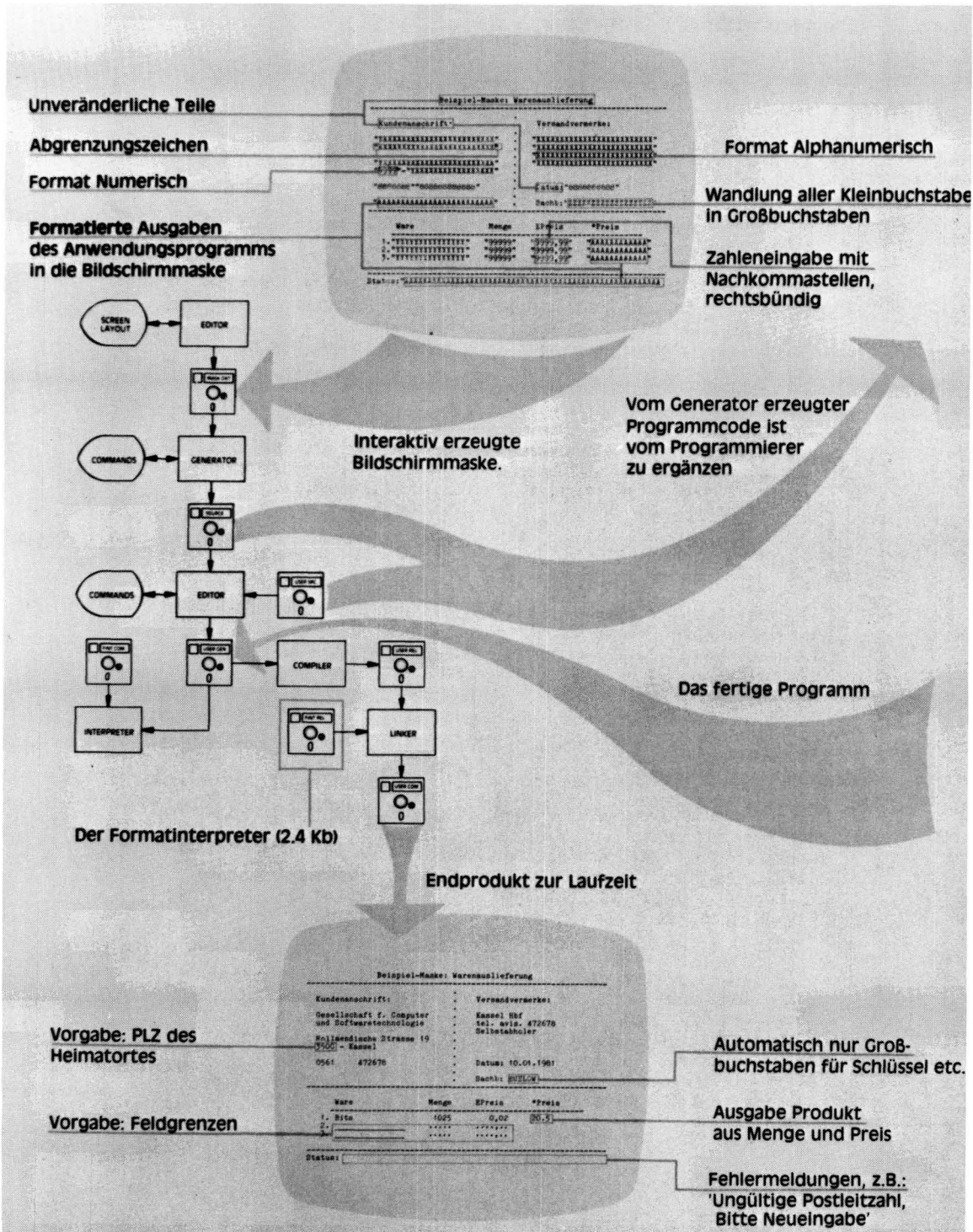
Das folgende Bild gibt noch einmal den Gesamtablauf wieder. Als Beispiel wurde eine typische Erfassungsmaske für Versandpapiere benutzt. Der erzeugte Programmcode ist PL/I.
(Hier folgt Seite 3 und 4 des Larve-Prospekts)

5. Schlußbetrachtung

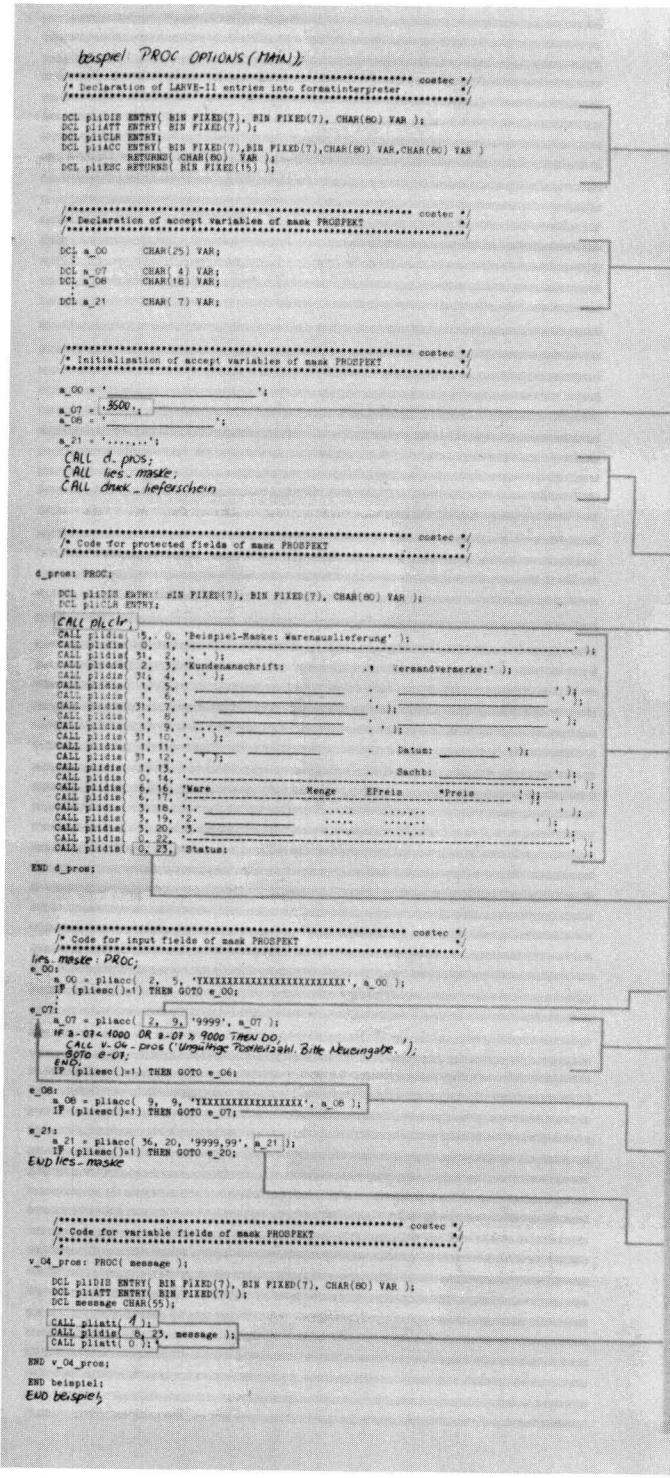
Das Werkzeug LARVE wurde ursprünglich in unserem Hause zur Rationalisierung unserer eigenen Softwareproduktion entwickelt. Der Formatinterpreter wurde in Assembler implementiert, der Maskengenerator in PL/I. Als Betriebssystemumgebung fand CP/M, ein verbreitetes Microcomputerbetriebssystem, Verwendung. Um jedoch den Übergang auf die 16bit-Generation und andere Betriebssysteme zu bewältigen, wird das gesamte Paket zur Zeit in C umgeschrieben.

Durch die Beschränkung der Formate und Funktionen auf die Grundbausteine ist LARVE universell einsetzbar. Bestehende Anwendungen bzw. Entwicklungen sind zum Beispiel Systemprogramme, Graphiksoftware, Lungenfunktionsmeßplatz, Einbestellung von Schildrüsenkranken und Steuerprogramme einer NC-Maschine.

Ablauf der automatischen Bild



schirmprogrammierung mit Larve:



```
beispiel: PROC OPTIONS (MAIN);
/*****************************************************  costec */
/* Declaration of LARVE-II entries into formatinterpreter      */
/*****************************************************/

   DCL pliDIE ENTRY( BIN FIXED(7), BIN FIXED(7), CHAR(80) VAR );
   DCL pliATT ENTRY( BIN FIXED(7) );
   DCL pliCLR ENTRY;
   DCL pliACC ENTRY( BIN FIXED(7),BIN FIXED(7),CHAR(80) VAR,CHAR(80) VAR )
              RETURNS( CHAR(80) VAR );
   DCL pliESC RETURNS( BIN FIXED(15) );


/*****************************************************  costec */
/* Declaration of accept variables of mask PROSPEKT            */
/*****************************************************/

   DCL a_00      CHAR(25) VAR;
   ...
   DCL a_07      CHAR( 4) VAR;
   DCL a_08      CHAR(18) VAR;
   ...
   DCL a_21      CHAR( 7) VAR;


/*****************************************************  costec */
/* Initialisation of accept variables of mask PROSPEKT         */
/*****************************************************/

   a_00 = '                         ';
   a_07 = '3500';
   a_08 = '                  ';
   ...
   a_21 = '.......';
   CALL d_pros;
   CALL lies_maske;
   CALL druck_lieferschein


/*****************************************************  costec */
/* Code for protected fields of mask PROSPEKT                  */
/*****************************************************/

d_pros: PROC;

   DCL pliDIE ENTRY( BIN FIXED(7), BIN FIXED(7), CHAR(80) VAR );
   DCL pliCLR ENTRY;

   CALL pliclr;
   CALL plidia( 5, 0, 'Beispiel-Maske: Warenauslieferung' );
   CALL plidia( 0, 1, '--------------------------------------------------------------------------------' );
   CALL plidia( 51, 2, '. ' );
   CALL plidia( 2, 3, 'Kundenanschrift:                .'  Versandvermerke:' );
   CALL plidia( 51, 4, '. ' );
   CALL plidia( 1, 5, '  _____                         .  _____                ' );
   CALL plidia( 1, 6, '  _____                         .  _____                ' );
   CALL plidia( 51, 7, '. ' );
   CALL plidia( 1, 8, '  _____                         .  _____                ' );
   CALL plidia( 1, 9, '  ____ ____                     .                       ' );
   CALL plidia( 51, 10, '. ' );
   CALL plidia( 1, 11, '  ____ ____                     .  Datum: _____        ' );
   CALL plidia( 51, 12, '. ' );
   CALL plidia( 1, 13, '                                .  Sachb: _____        ' );
   CALL plidia( 0, 14, '--------------------------------------------------------------------------------' );
   CALL plidia( 6, 16, 'Ware          Menge    EPreis     Preis                ' );
   CALL plidia( 6, 17, '--------------------------------------------------------------------------------' );
   CALL plidia( 5, 18, '1. _____      .....    .......                         ' );
   CALL plidia( 5, 19, '2. _____      .....    .......                         ' );
   CALL plidia( 5, 20, '3. _____      .....    .......                         ' );
   CALL plidia( 0, 22, '--------------------------------------------------------------------------------' );
   CALL plidia( 0, 23, 'Status:                                                ' );
END d_pros;


/*****************************************************  costec */
/* Code for input fields of mask PROSPEKT                      */
/*****************************************************/

lies_maske: PROC;
e_00:
   a_00 = pliacc( 2, 5, 'YXXXXXXXXXXXXXXXXXXXXXXXX', a_00 );
   IF (pliesc()=1) THEN GOTO e_00;
   ...
e_07:
   a_07 = pliacc( 2, 9, '9999', a_07 );
   IF a-07 < 4000 OR a-07 > 9000 THEN DO;
      CALL v_04_pros ('Ungültige Postleitzahl. Bitte Neueingabe.');
      GOTO e-07;
   END;
   IF (pliesc()=1) THEN GOTO e_06;

e_08:
   a_08 = pliacc( 9, 9, 'YXXXXXXXXXXXXXXXXX', a_08 );
   IF (pliesc()=1) THEN GOTO e_07;

e_21:
   a_21 = pliacc( 36, 20, '9999,99', a_21 );
   IF (pliesc()=1) THEN GOTO e_20;
END lies_maske


/*****************************************************  costec */
/* Code for variable fields of mask PROSPEKT                   */
/*****************************************************/

v_04_pros: PROC( message );

   DCL pliDIE ENTRY( BIN FIXED(7), BIN FIXED(7), CHAR(80) VAR );
   DCL pliATT ENTRY( BIN FIXED(7) );
   DCL message CHAR(55);

   CALL pliatt( 1 );
   CALL plidia( 8, 23, message );
   CALL pliatt( 0 );

END v_04_pros;

END beispiel;
END beispiel;
```

Verkürzte Darstellung der Generatorausgabe (Ergänzungen des Programmierers handschriftlich).

Deklaration der Einsparungspunkte in den Formatinterpreter für den Linker.

Automatische Deklaration der Variablen in der entsprechenden Länge. Die Variablen können gemäß ihrer Aufgabe umbenannt werden: Zum Beispiel a-07 in PLZ etc.

Startwert z.B. Heimatpostleitzahl, da häufig benötigt. Dieser Wert erscheint dann auf dem Bildschirm und kann sofort übernommen werden. Oder ediert, zum Beispiel 3501, 3512, 8500 etc.

Steuerleiste: Aufbau der Maske, Eingabe der Werte und Ausdruck mit anderem Programmteil.

Löschen des Bildschirms (Terminalabhängig!)

Feste Bestandteile der Bildschirmmaske, die noch nach Belieben von Hand verändert werden können.

Koordinaten der Ausgabe (Cursorpositionierung Terminalabhängig!)

Koordinaten der Eingabe

Semantische Prüfung der Variable a-07 (PLZ) auf Bereichsüberschreitung.

Automatische Rückwärtsverkettung. So kann jedes Eingabefeld der Maske nachträglich für Korrekturen erreicht werden.

Startwert der Eingabe.

Ausgabe der Fehlermeldung mit **logischem** Attribut 1, danach Attribut wieder ausgeschaltet.
Realisation des Attributs (Blinken, Highlight etc.) durch Konfigurationsprogramm, also nach Geschmack und Terminal variierbar.

Neben einer drastischen Aufwandreduktion bei der Erstellung von Dialogprogrammen führt es durch die Befreiung des Programmiers von Ein/Ausgabe-Details zu kreativerer Ausnutzung des Mediums Bildschirm. Die Portabilität von LARVE eröffnet für die Erstellung von Dialogstandardsoftware auf Microcomputern neue Perspektiven, die auf Großrechnern zum Teil bereits Standard sind.

Literatur:

/1/ Heinrich, Lutz: Computerleistung am Arbeitsplatz
München: Verlag Oldenbourg (1978)

/2/ Martin, James: Design of Man-Computer Dialogues
Englewood Cliffs, N.J.:
Prentice Hall (1973)

Lothar B. Ehrhardt / Wolfgang K. Weber
costec Gesellschaft für Computer-
und Softwaretechnologie mbH
Holländische Straße 19

3500 Kassel