DruckNT - Schulung

Version 2.30

Erstellt: 12.02.2001

Inhaltsverzeichnis

1. Allgemeines

In dieser Beschreibung werden die technischen Grundlagen des Druckmoduls der Lexware aufgezeigt und eine Einführung in die Scriptsprache zur Erstellung von Formularen gegeben.

Diese Schulung wurde erstellt, als das Druckmodul in der ersten Version verfügbar war. Mittlerweile sind einige neue Befehle zur Formularsprache hinzugekommen. Da die vorliegende Schulung eine didaktische Einführung sein soll, sind hier nicht alle neuen Befehle verzeichnet, dazu dient die Onlinehilfe, die vom Formulareditor aus aufgerufen werden kann.

2. Grundlagen zum DruckNT Formulareditor

2.1 Die Dateien eines DruckNT Formulares

Das Druckmodul arbeitet mit folgenden Dateien:

Scriptdateien

(*.LSF; **L**exware **S**cript **F**ile)

Interfacedateien

(*.INT; **Int**erface File)

Bei den Script- und Interfacedateien handelt es sich um textbasierte Dateien, die mit dem Lexware-Formulareditor erstellt und bearbeitet werden können. Da es sich um reine Textdateien handelt, können diese Dateien grundsätzlich mit jedem Editor bearbeitet werden, diese Dateien enthalten alle Elemente (Anweisungen, Funktionen, Schleifen, Tabellen, ...) die einen Ausdruck beschreiben, wobei in den Interfacedateien bevorzugt Funktions- und Konstantendefinitionen und Funktionen, die von mehreren Formularen verwendet werden, enthalten sind.

Ausführbare Formulardateien

(Binärdateien: *.VMB; **V**irtual **M**achine **B**inary File)

Die ausführbaren Binärdateien (-> VMB-Datei) sind die Dateien, die vom Druckmodul ausgeführt werden. Eine VMB-Datei kann aus mehreren Script- und Interfacedateien mit dem Formulareditor erzeugt werden. Da die Formulardateien in binärer Form vorliegen, können diese vom Druckmodul schneller ausgeführt werden, als eine Textdatei, die erst interpretiert werden muss.

Druckereinstellungsdateien

(*.LSP)

Diese Dateien speichern spezielle Druckereinstellungen, die in einem Formular über den SetPrintDevice-Befehl aktiviert werden können.

Das Konzept von DruckNT erlaubt es, mehrere Dateien zu einem ‚Projekt' zusammenzufügen. Somit können bestimmte Eigenschaften oder Funktionen einmalig in einer zentralen Script- oder Interfacedatei beschrieben werden und in mehreren Formularen verwendet werden.

Besondere Interfacedateien

DruckNT.INT

In dieser Datei werden alle vom Druckmodul zur Verfügung gestellten Funktionen und Konstanten definiert. Diese Datei wird für jedes Formular benötigt.

LexVM.INT

Diese Datei enthält Funktionen und Konstanten der virtuellen Maschine.

<App>.INT

Für jede Lexware Applikation existiert eine spezielle Interfacedatei (z.B. bs.INT für das Lexware bsRechnungswesen), in der alle applikationsspezifischen Konstanten und die vom Programm zur Verfügung gestellten Schnittstellenfunktionen definiert werden.

2.2 Der Formulareditor

DNTDev23.EXE (**D**rucken **NT Dev**elopment Editor)

2.21 Starten des Formulareditors

Wenn Sie den Formulareditor starten möchten, gehen Sie wie folgt vor:

1. Starten Sie den Windows Explorer. Danach wechseln Sie in das Verzeichnis in welchem Ihr Lexware Programm installiert ist.

2. Jetzt suchen Sie in diesem Verzeichnis nach DNTDev23.exe. Es ist auch möglich, dass die DNTDev23.exe in einem Unterverzeichnis liegt. Wenn Sie die EXE gefunden haben, starten Sie diese mit einem Doppelklick.

Wenn Ihnen nicht bekannt ist, wo Ihr Lexware Programm installiert ist, dann starten Sie das Programm und klicken dann auf das **?** und von da auf **Info**.

In der Info-Box drücken Sie auf die Schaltfläche **Systeminfo**.

Verschiedene Informationen zum Programm werden hiermit aufbereitet und Sie können die gewünschten Informationen entnehmen.

2.22 Info zum Formulareditor

Im Anhang befindet sich ein Schritt für Schritt Kurs, mit dem Sie den Umgang und die Funktionsweise des Druckmoduls erlernen können.

Über den Formulareditor kann eine ausführbare Formulardatei (VMB-Datei) aus den entsprechenden Script- und Interfacedateien erzeugt (-> kompiliert) werden. Da sich eine VMB-Datei aus mehreren Script- und Interfacedateien ergeben kann, muss im Formulareditor eine LSF-Datei als Hauptformular (-> Projekt) über den Menüpunkt „*Projekt-Formular“* gesetzt werden. Der Name des aktuellen Projektes wird in der Titelzeile linksbündig angezeigt. Über den Menüpunkt „*Projekt-Kompilieren“* kann die zugehörige VMB-Datei erzeugt werden. Im Ausgabefenster wird der Status der Übersetzung angezeigt. Falls ein Syntaxfehler aufgetreten ist, wird dieser unter Angabe der Zeilennummer

angegeben. Wenn das Projekt erfolgreich übersetzt werden konnte, werden die allgemeinen Projektangaben und die Versionsnummer des Übersetzers angezeigt. Die VMB-Datei erhält den gleichen Dateinamen wie das Hauptformular und die Dateierweiterung VMB. Über den Menüpunkt „*Projekt-Formular“* können auch mehrere Hauptformulare definiert werden (die Mehrfachauswahl erfolgt entsprechend den Windowskonventionen mit der <Feststell>- oder der <Strg> Taste). In diesem Fall werden beim Kompilieren alle ausgewählten Formulare erstellt, wobei der Compiler nach einer fehlerhaften Datei abbricht. Die Mehrfachauswahl ist besonders dann nützlich, wenn in einer Interfacedatei, die von mehreren Formularen verwendet wird, eine Änderung vorgenommen wurde.

Die Seitenvoransicht über den Menüpunkt „*Projekt-Ausführen“* steht nur dann zur Verfügung, wenn im aktiven Projekt keine applikationsspezifischen Funktionen verwendet werden.

Während des Bearbeitens einer Datei kann über die <F1>-Taste kontextbezogen Hilfe zu den Schlüsselwörtern, Funktionen, Konstanten und Datentypen angefordert werden, sobald sich die Schreibmarke innerhalb eines entsprechenden Begriffs befindet.

Zur besseren Lesbarkeit werden folgende Ausdrücke und Bereiche im Editor unterschiedlich farblich dargestellt (die Bedeutung der einzelnen Begriffe werden unter dem Abschnitt ´Die Scriptsprache für Formulare, Elemente einer Scriptdatei´ näher erläutert):

Farbe	Ausdruck/Bereich
Grün	Kommentare
Rot	Texte (Strings in Anführungszeichen)
Hellblau	Schlüsselwörter
Dunkelblau	Funktionen
Violett	Konstanten, Datentypen

2.23 Technische Info

Die Farbdefinition und die Textzuordnung der Schlüsselwörter, Funktionen, Konstanten und Datentypen erfolgt in der Datei LXSCRIPT.INI, die sich im gleichen Verzeichnis wie der Formulareditor befinden muss. Die Farbdefinitionen können jederzeit geändert oder erweitert werden.

3. Die Skriptsprache

In diesem Kapitel werden die für die jeweilige Verwendung zur Verfügung stehenden Funktionen im entsprechenden Zusammenhang aufgeführt, eine genaue Beschreibung der Parameter erfolgt hier jedoch nicht. Diese Beschreibungen sind der Sprachreferenz über die Onlinehilfe des Formulareditors zu entnehmen.

3.1 Die Syntax

Die Lexware Scriptsprache von DruckNT lehnt sich stark an die Syntax von BASIC an.

Generell wird bei allen Anweisungen keine Unterscheidung zwischen Groß- und Kleinschreibung gemacht. Um die Lesbarkeit zu verbessern, sollte jedoch eine einheitliche Verwendung von Groß- und Kleinschreibung innerhalb eines Formulares verwendet werden.

3.11 Kommentare

Innerhalb einer Scriptdatei können an jeder beliebigen Stelle Kommentare eingefügt werden, indem ein Hochkomma vorangestellt wird. Ab einem Hochkomma wird der Rest einer Zeile als Kommentar interpretiert. Es sollte möglichst viel von Kommentaren Gebrauch gemacht werden, um die Lesbarkeit und Reproduzierbarkeit des Skriptes zu gewährleisten.

Am Anfang einer Scriptdatei sollte als Kommentar ein Formularkopf enthalten sein, aus dem eine kurze Beschreibung der Datei, der Autor und nach Möglichkeit das Datum der letzten Änderung hervorgeht.

Hier ein Beispiel:

```
'/*****************************************************************************/
'/*** Lexware bsRechnungswesen                                             ***/
'/*****************************************************************************/
'/*** Dateiname:ustverpr.lsf                                               ***/
'/*** zugrundeliegende Datei:          -                                   ***/
'/*** Ersteller:RalfL                                                      ***/
'/*** Erstellt am:27.01.99                                                 ***/
'/*** Beschreibung:          Umsatzsteuerverprobung Detailliert,           ***/
'/***                       sortiert nach Steuerkürzel                    ***/
'/*** Kommentar:                        -                                 ***/
'/*****************************************************************************/
```

3.12 Einbinden von Interfacedateien

Mit dem Include-Befehl werden die für das Formular benötigten Interfacedateien eingebunden.

An den Stellen im Formular, an denen ein Include Befehl steht, wird beim Übersetzen des Formulars der Quellcode der entsprechenden Interfacedatei eingefügt. Das bedeutet, dass die Reihenfolge der Dateien, die mit Include eingebunden werden, eine wichtige Rolle spielen kann.

Beispiel: Werden in einer Interfacedatei, die mit Include eingebunden wird, applikationsspezifische Konstanten und die vom Programm zur Verfügung gestellten Schnittstellenfunktionen definiert, und es erfolgt vor der Includezeile ein Zugriff auf eine dieser Konstanten bzw. Funktionen, erhält man beim Übersetzen des Formulars eine Fehlermeldung.

Der Include-Befehl ermöglicht aber nicht nur das Einbinden von Interfacedateien sondern auch von anderen Formulardateien (*.LSF). Ein Formular kann so aus einzelnen Komponenten aufgebaut werden, die wiederum in mehreren Formularen eingebunden sind. Ein typisches Beispiel für eine derartige Komponente ist der Kopf oder Fuß einer Seite, der so nur einmal erstellt werden muss und dann in jedem Ausdruck verwendet werden kann und immer gleich aussieht.
Von der Möglichkeit, LSF-Dateien einzubinden, sollte jedoch nur in Ausnahmefällen Gebrauch gemacht werden, da dies die Formulare nicht sehr leserlich macht und die Fehlersuche erheblich erschweren kann.

```
 Include "DruckNT.int" ' Interface-Deklarationen der DruckNT-
Bibliothek
 Include "bs.int" ' Applikationsspezifische-Deklarationen
 Include "Kopf.lsf" ' LSF-Datei mit Deklarationen für die Kopfzeile
```

Hinweis: Die Interfacedateien "DruckNT.int" und "LexVM.int" enthalten die Deklarationen aller standarmäßig vorhandenen Befehle der Formularsprache. Sie sind daher an die aktuelle Version des Formulareditors gebunden und werden immer aus dessen Dateiverzeichnis eingebunden. Sie sollten nicht verändert werden!

3.13 Variablen und Datentypen

Eine Variable stellt einen Platzhalter für einen Wert eines bestimmten Datentyps dar. Eine Variable ist durch einen eindeutigen Namen (auch Identifikator oder Bezeichner) und einen bestimmten Datentyp gekennzeichnet. Bevor innerhalb einer Scriptdatei eine Variable verwendet werden kann, muss diese zuerst definiert werden. Die Lexware Scriptsprache stellt folgende Grunddatentypen zur Verfügung:

String	Text bzw. beliebige Zeichenkette
Numeric	Numerischer Wert (Ganzzahl, Kommazahl, Datum/Uhrzeit)
Bool	Wahrheitswert (wahr (TRUE) oder falsch (FALSE))

Zur Definition einer Variablen werden dieSchlüsselwörter '*Dim*' und '*As*' verwendet:

```
Dim strText As String
Dim nZahl   As Numeric
Dim bOK     As Bool
```

Eine Variable, die innerhalb einer Funktion definiert wurde, ist nur innerhalb dieser Funktion gültig. Alle anderen Variablen sind global gültig und können somit auch in einer Funktion angesprochen werden. Zwar werden bei gleichnamigen Variablen, die global und innerhalb einer Funktion definiert sind, die innerhalb der Funktion definierten vorrangig bearbeitet, dennoch sollten Sie nach Möglichkeit Doppelbelegungen vermeiden.

Neben Variablen können auch Konstanten deklariert werden. Im Gegensatz zu einer Variablen, deren Inhalt sich während der Laufzeit eines Formulars jederzeit durch verschiedene Verarbeitungen verändern kann, repräsentiert eine Konstante einen festen Wert, der zur Laufzeit durch keine Anweisung modifiziert werden kann.

```
Const strTEXT As "Beispieltext"
Const nZAHL   As 100
```

3.14 Ausdrücke, Wertzuweisungen, Operatoren

Ein Ausdruck ist eine Formel zur Berechnung eines Wertes. In einem Ausdruck werden konstante Werte, Variablen und Ergebnisse von Funktionen mit Hilfe von Operatoren verknüpft. Innerhalb eines Ausdrucks dürfen nur Werte des gleichen Datentyps verwendet werden und diese dürfen nur mit Operatoren verknüpft werden, die für diesen Datentyp zulässig sind.

Mit einer Wertzuweisung wird einer Variablen ein bestimmter Wert zugewiesen. Dieser Wert kann das Ergebnis einer Funktion oder eines Ausdruckes sein. Der Typ der Wertzuweisung muss dem Datentyp der Variablen entsprechen. Um bei Berechnungen die Reihenfolge der Bearbeitung zu bestimmen, können Klammern verwendet werden.

Folgende Operatoren werden unterstützt:

Stringoperatoren

Operator	Funktion	Ergebnistyp
+	Hängt zwei Strings aneinander.	String.
=	Vergleicht zwei Strings	Bool
<>	Vergleicht zwei Strings auf Ungleichheit	Bool

Numerische Operatoren

Operator	Funktion	Ergebnistyp
+	Addition	Numeric
-	Subtraktion	Numeric
*	Multiplikation	Numeric
/	Division	Numeric
=	Gleich	Bool
<	Kleiner	Bool
>	Größer	Bool
<>	Ungleich	Bool
<=	kleiner oder gleich	Bool
>=	größer oder gleich	Bool

Bool‘sche Operatoren

Operator	Funktion	Ergebnistyp
And	Logisches "und"	Bool
Or	Logisches "oder"	Bool
Not	Logische Negation	Bool

```
´ Beispiel 1
Dim nNr0 As Numeric
Dim nNr1 As Numeric
Dim x    As Numeric

nNr0 = 10
nNr1 = 5.5
x = nNr0 + (nNr1 * 100 )

´ Beispiel 2
Dim strVorName  As String
Dim strNachName As String
Dim strName     As String

strVorName = "James"
strNachName = "Bond"
strName = strVorName + " " + strNachName

´ Beispiel 3
Dim bOk  As Bool
Dim nMin As Numeric
Dim nAkt As Numeric

nMin = 10
nAkt = 20
bOk = nAkt >= nMin
```

3.2 Ausgabefunktionen

3.21 Ausgabe von Text

Zur Ausgabe von Text stehen folgende Funktionen zur Verfügung:

DrawText(Text as String)
DrawTextLn(Text as String)
DrawTextExt(x as Numeric, y as Numeric, Text as String)

Die beiden ersten Funktionen geben den übergebenen Text an der aktuellen Ausgabeposition aus, wobei die Funktion *DrawTextLn()* nach der Textausgabe einen Zeilenvorschub an den Anfang der nächsten Zeile vornimmt. Mit der dritten Funktion kann die Position des auszugebenden Textes bezogen auf die eingestellten Ränder links und oben bestimmt werden. Die Positionsparameter haben die Einheit 0,1 mm und werden ab dem eingestellten Seitenrand gerechnet. Da die Funktionen zur Textausgabe sehr häufig benötigt werden, existieren folgende Kurzformen:

T(Text as String)	entspricht DrawText(Text as String)
TL(Text as String)	entspricht DrawTextLn(Text as String)
LF	entspricht DrawTextLn(“”), bewirkt also einen Zeilenumbruch ohne Textangabe

```
Dim strText As String

strText = "Beispieltext"

DrawText( "Ausgabe eines " + strText ) ´ohne Zeilenumbruch
bzw.
T( "Ausgabe eines " + strText ) ´ohne Zeilenumbruch

DrawTextLn( "Ausgabe eines " + strText ) ´mit Zeilenumbruch
bzw.
TL( "Ausgabe eines " + strText ) ´mit Zeilenumbruch

´Gibt den angegebenen Text 1 cm links und 2 cm ab dem eingestellten
´Ausgabebereich (z.B. SetBodyMargin())
DrawTextExt( 100, 200, "Ausgabe eines " + strText )

LF ´Fügt eine Leerzeile mit Zeilenumbruch ein
```

Bei allen Funktionen zur Ausgabe von Text (-> String) kann der auszugebende Text durch die weiter oben aufgeführten Stringoperatoren aus mehreren Einzeltexten oder aus Ergebnissen von Funktionen, die einen String als Ergebnistyp liefern, zusammengestellt werden. Ein Zeilenumbruch wird durch die Zeichen ´\n´ ausgelöst. Ob vom Druckmodul ein automatischer Zeilenumbruch bei der Ausgabe von längerem Text durchgeführt werden soll, kann mit der Funktion

SetWordWrap(Wrap as Bool)

bestimmt werden. Standardmäßig ist der automatische Zeilenumbruch ausgeschaltet.

3.22 Ausgabe von Zahlen, Beträgen, Datums- und Uhrzeitwerten

Die Ausgabe von numerischen Werten erfolgt über die selben Funktionen wie die Textausgabe. Dazu müssen die Werte jedoch zunächst als Text formatiert werden. Hierfür stehen folgende Funktionen zur Verfügung:

FormatNumeric(Format as String, n as Numeric) as String
FormatDate(Format as String, Date as Numeric) as String

Die Art der Formatierung kann jeweils über einen Formatstring, der an die Formatierungsfunktion übergeben wird, bestimmt werden. Der Format String kann beliebige Zeichen beinhalten. Der Numeric-Wert wird an einer bestimmten Stelle im String plaziert. Diese Stelle können Sie im Formatstring mit der Zeichenfolge %f bestimmen. Weitere Parameter in dieser Zeichenfolge bestimmen die Formatierung der Zahl.

Eine genauere Beschreibung dieser Formatstrings ist unter der jeweiligen Funktionsbeschreibung in der Onlinehilfe zu entnehmen.

```
Dim nBetrag As Numeric
nBetrag = 5.523
 TL( FormatNumeric( "Betrag: %.2f", nBetrag ) ) ´Gibt den Betrag mit
zwei Nachkommastellen aus
=>  Betrag: 5.52
```

Anmerkung:

Numericwerte besitzen immer sechs Nachkommastellen, mit denen auch die Berechnungen durchgeführt werden. Wenn man die Ausgabe-formatierung eines Wertes nun auf z.B. zwei Stellen nach dem Komma festlegt, erfolgt für die Ausgabe eine Rundung der übrigen, nicht auszugebenden Nachkommastellen (Zahlen von 1 bis 4 werden ab- und Zahlen von 5 bis 9 aufgerundet).
Dies kann in manchen Fällen bei der Ausgabe zu scheinbar falschen Werten führen.

Ein Beispiel soll dies verdeutlichen:

Es sollen folgende drei Zahlen addiert werden, 2,644, 1,234 und 1,004. Die Ausgabeformatierung ist auf zwei Stellen nach dem Komma eingestellt.

Sichtbare Ausgabe	Die Berechnung
2,64 1,23 1,00	2,64 | 4000 1,23 | 4000 1,00 | 4000
dies ergäbe eine Summe von **2,87**.	da aber mit allen Nachkommastellen gerechnet wird, ergibt dies eine Summe von **2,882000** bzw. die Ausgabe wäre **2,88**.

Die Formatierung von Zahlenwerten kann zusätzlich mit der Funktion

SetFormatNumericOptions(mode as Numeric, dezimaltrennzeichen as String, tausenderseperator as String)

beeinflusst werden. Mit dieser Funktion kann das Zeichen für den Tausenderseparator und das Dezimaltrennzeichen gesetzt werden. Neben der Standardformatierung (ein ´.´ als Dezimaltrennzeichen und kein Tausenderseparator) können entweder die Einstellungen aus der Systemsteuerung für Betragswerte übernommen oder die beiden Zeichen direkt übergeben werden.

```
 ´ Setzt die Standardeinstellungen
 SetFormatNumericOptions( FN_DEFAULT, "", "")
´ Setzt die Benutzer definierten Einstellungen
 SetFormatNumericOptions( FN_CUSTOM, ",", ".")
```

```
 ´ Setzt die Einstellungen aus der Systemsteuerung
 SetFormatNumericOptions( FN_LOCALE, "", "")
```

Für die Ausgabe von Datums- und Uhrzeitwerten wird kein spezieller Datentyp verwendet. Es wird dafür ein Wert des Typs *Numeric* verwendet. Auch Datums- und Uhrzeitwerte, die von einer Applikation übergeben werden, sind als Numeric angegeben.

```
´Gibt das aktuelle Datum mit Uhrzeit aus
TL( FormatDate( "%A, den %d.%B.%Y um %H:%M Uhr", Now() )
=>  Montag, den 13. September 1999 um 15.30 Uhr
```

3.23 Aktuelles Datum und Uhrzeit

Über die Funktion:

```
Now()
```

kann das aktuelle Datum und die Uhrzeit ermittelt werden. Der Rückgabewert dieser Funktion ist vom Typ *Numeric* und kann mit der Funktion *´FormatDate() ´* als Text formatiert werden.


```
´Gibt das aktuelle Datum mit Uhrzeit aus
TL( FormatDate( "%A, den %d.%B.%Y um %H:%M Uhr", Now() )
=>  Montag, den 13. September 1999 um 15.30 Uhr
```


3.24 Schriftarten

Die Funktionen zur Ausgabe von Text verwenden die aktuell gesetzte Schriftart. Die aktuelle Schriftart (-> ´Font´) wird mit der Funktion

SelectFont(id as FontType)

gesetzt. Dazu benötigt diese Funktion als Parameter einen Wert des Typs *FontType*.

Eine Variable des Typs *´FontType´* wird wiederum mit der Funktion

CreateFont(font as String, size as Numeric, type as Numeric, color as Numeric) as FontType

erzeugt. Ein Font wird durch die Schriftart, die Schriftgröße, den Schrifttyp (Normal, Fett, Kursiv) und die Schriftfarbe definiert. Für den Schrifttyp und die Farbe können Konstanten, die in DruckNT.INT definiert sind, verwendet werden. Diese sind in der Onlinehilfe beschrieben.
Mit diesen Funktionen können alle im System installierten Schriftarten erzeugt werden.

Innerhalb einer Formulardatei ist es zu empfehlen, alle im Formular benötigten Schriftarten am Anfang der Datei zu definieren und zu erzeugen. Dadurch wird es später einfacher, die Schriftarten eines Formulares individuell anzupassen.

```
 ´ notwendige Variablen deklarieren
 Dim ARIAL_10        As FontType
 Dim ARIAL_FETT_10   As FontType
 Dim ARIAL_KURSIV_10 As FontType
 ´ Font für Fett-, Normal- und Kursivschrift erzeugen
 ARIAL_10        = CreateFont( "Arial", 10, FONT_NORMAL,
COLOR_BLACK )
 ARIAL_FETT_10   = CreateFont( "Arial", 10, FONT_BOLD, COLOR_BLACK )
 ARIAL_KURSIV_10 = CreateFont( "Arial", 10, FONT_ITALIC,
COLOR_BLACK )
 ´ die aktuelle Schrift setzen
 SelectFont( ARIAL_10 )
```

Eine weitere Möglichkeit, innerhalb eines Formulars die Schriftart zu wechseln, ist mit der Funktion

SetFont(font as String, size as Numeric, type as Numeric, color as Numeric)

```
´aktuelle Schriftart setzen
SetFont( "Arial", 10, FONT_NORMAL, COLOR_BLACK )
```

Bei dieser Funktion können <u>keine</u> zuvor in Variablen definierten Schriftarten verwendet werden. Dafür gibt es weitere spezielle Funktionen um die Schriftart, die Schriftgröße, den Schrifttyp (Normal, Fett, Kursiv) und die Schriftfarbe zu ändern.

SetFontName(font as String)
SetFontSize(size as Numeric)
SetFontType(type as Numeric)
SetFontColor(color as Numeric)

Gewechselt wird nur, je nach aufgerufener Funktion, der übergebene Parameter. Die übrigen Attribute der Schrift, wie z.B. Farbe und Typ, bleiben erhalten.

3.25 Tabulatoren

Zur Positionierung von Text kann mit Tabulatoren gearbeitet werden.

Die zu setzenden Tabulatoren werden in einer Zeichenfolge getrennt durch Leerzeichen angegeben. Ein Tabulator wird mit Hilfe von zwei Eigenschaften beschrieben. Die erste beschreibt die Ausrichtung und die zweite die horizontale Position. Die Positionsparameter haben die Einheit 0,1 mm. Folgende Ausrichtungen werden berücksichtigt:

L - linksbündig
R - rechtsbündig
C - zentriert

Die Positionierung folgt direkt auf das Ausrichtungszeichen. Existiert für einen zentrierten oder rechtsbündigen Tabulator keine Positionsangabe, so wird als Position die Seitenmitte bzw. der rechte Seitenrand definiert. Eine negative rechtsbündige Tabulatorposition bedeutet, dass die Position vom rechten Seitenrand berechnet werden soll. Ein Tabulator wird innerhalb einer Textzeile durch die Zeichen *´\t´* gesetzt. Um die Lesbarkeit eines Formulares zu erhöhen, ist es zu empfehlen, eine Konstante für den Tabulator zu definieren und bei der Textausgabe zu verwenden.

Tabulatoren werden mit den Funktionen

SetTabs(tabs as String)
RemoveAllTabs()

gesetzt bzw. wieder zurückgesetzt. Einmal gesetzte Tabulatoren sind solange gültig, bis sie wieder einzeln oder komplett mit der entsprechenden Funktion zurückgesetzt werden.

```
´ Tabulatorzeichen als Konstante definieren
Const Tab As "\t"
```

```
´linksbündige Tabulatoren auf 4cm und auf 10cm setzen
SetTabs( "L400 L1000" )
TL( "\tArtikelbezeichnung\tEinzelpreis" )
RemoveAllTabs() ´ Tabs löschen
´zweiten Tab auf rechtsbündig setzen
SetTabs( "L400 R1000" )
TL( Tab + strArtikel + Tab + nBetrag )
```

3.26 Aktuelle Ausgabeposition

Abgesehen von der Funktion *´DrawTextExt()´*, bei der die Position der Ausgabe direkt an die Funktion übergeben wird, erfolgt die Textausgabe grundsätzlich an der aktuellen Ausgabeposition. Dabei wird der Text rechts unterhalb der Position ausgegeben. Die Ausgabeposition ist nach Aufruf der Textausgabefunktionen, die keinen Zeilenvorschub ausführen, direkt hinter dem zuletzt ausgegebenen Text. Wird ein Zeilenvorschub ausgeführt, so ist die Position am linken Rand des aktuellen Druckbereiches jeweils um die Zeilenhöhe der aktuellen Schriftart nach unten verschoben. Die weiter unten beschriebenen Funktionen zur Ausgabe von Linien beeinflussen die aktuelle Ausgabefunktion ebenfalls. Zum besseren Verständnis stellen Sie sich den Druckbereich in einem Formular als Koordinatensystem vor. Die aktuelle Ausgabeposition kann mit der Funktion

SetPos(x as Numeric, y as Numeric)

direkt gesetzt werden. Zur Ermittlung der aktuellen Position stehen die Funktionen

GetPosX() as Numeric
GetPosY() as Numeric
GetPos(x ref Numeric, y ref Numeric)

zur Verfügung. Alle Werte beziehen sich immer auf die obere, linke Ecke des aktuellen Ausgabebereichs und **nicht** auf den Seitenrand (näheres hierzu weiter unten). Die eingestellten Ränder müssen also berücksichtigt werden.

```
Dim xPos As Numeric
Dim yPos As Numeric
´Ausgabeposition auf 5 cm von oben setzen
SetPos( 0, 500 )
```

```
 TL( "Beispieltext" )
 ´die x- und y-Position der aktuellen Ausgabeposition getrennt
ermitteln
 xPos = GetPosX()
 yPos = GetPosy()
 ´beide Positionen mit nur einem Funktionsaufruf ermitteln
 GetPos( xPos, yPos )
```

3.27 Bedingungen und Schleifen

Eine bedingte Programmausführung wird erreicht durch die Schlüsselwörter

If - Then, Else und *End If*

Es besteht darüber hinaus die Möglichkeit, für eine als Bool'schen Ausdruck anzugebende Bedingung einen beliebigen Block von Anweisungen auszuführen und optional auch einen beliebigen anderen Block auszuführen, falls die Bedingung nicht erfüllt ist.

```
 Dim bNegativ As Bool
 Dim n        As Numeric

 n = 2
 If n < 0 Then
    bNegativ = TRUE
 Else
    bNegativ = FALSE
 End If
```

Wenn ein beliebiger Anweisungsblock mehrmals ausgeführt werden soll, so kann dies über die Schlüsselwörter

While – Do und *End While*

realisiert werden. Der entsprechende Block wird solange wiederholt, bis die angegebene Bedingung nicht mehr erfüllt ist.

```
 Dim nBetrag   As Numeric
 Dim nVolleDM  As Numeric
```

```
nBetrag = 34.37
nVolleDM = 0

While nBetrag > 1 Do
   nVolleDM = nVolleDM + 1
   nBetrag = nBetrag - 1
End While
```

Achtung: Es muss sichergestellt sein, dass die Schleife irgendwann beendet wird. Ist dies nicht der Fall, dann handelt es sich um eine Endlos-Schleife und Sie können beim Aufbereiten des Ausdrucks nur noch die Schaltfläche „Abbrechen“ betätigen.

Wie in den beiden vorangegangenen Beispielen ersichtlich ist, wurde der Anweisungsblock für die unterschiedlichen Bedingungen jeweils um einen Tabulator eingerückt. Da es erlaubt ist, Bedingungen und Schleifen beliebig zu verschachteln, wird durch das Einrücken jedes Anweisungsblockes die Lesbarkeit wesentlich erhöht, da klar erkennbar ist, wo der jeweilige Block beginnt und endet. Der Formulareditor stellt nach einem Zeilenumbruch den Cursor automatisch auf den gleichen Einzug wie die vorherige Zeile.

In den Fällen, in denen sich ein Anweisungsblock (auch zusammen mit weiteren, verschachtelten Blöcken) soweit ausdehnt, dass Blockanfang und –ende nicht zusammen am Bildschirm sichtbar sind, ist es sinnvoll, hinter der Endemarke des Blockes einen entsprechenden Kommentar einzufügen, damit klar ersichtlich ist, welcher Block hier endet. Ebenso sollte am Anfang eines Blockes eine kurze Beschreibung erfolgen.

```
If PersonenKonto() Then
   ´ Bei einem Personenkonto die Adressangaben ausgeben
   ...
   ...
End If ´ Ende Personenkonto
```

3.3 Funktionen

3.31 Funktionen

Funktionen bilden eine abgeschlossene Einheit in einem Formular. So lässt sich ein Formular aus mehreren Funktionen zusammenbauen und damit modularisieren.
Eine Funktion führt eine bestimmte Aktion aus. Im Allgemeinen benötigt eine Funktion eine oder mehrere Angaben (-> Parameter) zum Ausführen dieser

Aktion und liefert einen Wert als Ergebnis (->Rückgabewert, Returnwert) zurück.

Wird eine Funktion aufgerufen, so kann für die einzelnen Parameter entweder eine Variable des entsprechenden Datentyps oder eine direkte Wertzuweisung erfolgen.

Bei den Funktionen wird zwischen externen Funktionen, die vom Druckmodul oder der aufrufenden Applikation zur Verfügung gestellt werden, und internen Funktionen unterschieden, die im aktuellen Script direkt enthalten sind.

Externe Funktionen werden über das Schlüsselwort *´Declare external´* in einer Interfacedatei definiert. Alle vom Druckmodul zur Verfügung gestellten Funktionen sind in der Datei DruckNT.INT enthalten. Wichtig!!! „An dieser Stelle muss nochmals erwähnt werden, dass die Datei DruckNT.INT in jedem Formular enthalten sein muss, da dort alle Funktionen und Konstanten zur Verfügung gestellt werden."

Interne Funktionen werden innerhalb einer Script- oder Interfacedatei über das Schlüsselwort *´Function´* definiert. Die Beschreibung der Parameter und des Rückgabewertes ist in beiden Fällen gleich.

Die Parameterliste

Die Beschreibung erfolgt durch Angabe von Parametername und Datentyp entsprechend einer Variablendefinition, jeweils durch Kommata voneinander getrennt innerhalb einer Klammer. Als spezielle Parameter gibt es hier die sogenannten Referenzparameter. Diese werden verwendet, wenn eine Funktion mehr als einen Rückgabewert als Ergebnis zurückliefert. Wird anstelle des Schlüsselwortes *´As´* zwischen Parametername und Datentyp das Schlüsselwort *´Ref´* verwendet, so darf beim Aufruf der entsprechenden Funktion dieser Parameter nur in Form einer Variablen des entsprechenden Datentyps übergeben werden.
Wird eine Variable an einen Referenzparameter übergeben, dann enthält die so übergebene Variable nach dem Funktionsaufruf den Wert, den der entsprechende Parameter innerhalb der Funktion hatte. Dadurch ist es möglich, dass eine Funktion mehrere Rückgabewerte an den Aufrufer übergibt.

Der Rückgabewert

Der Datentyp des Rückgabewertes wird entsprechend dem Datentyp einer Variablen über das Schlüsselwort *´As´* angegeben.

Benötigt eine Funktion keine Parameter, so entfällt die Parameterliste innerhalb der Klammern. Liefert eine Funktion keinen Rückgabewert, so entfällt die komplette Angabe des Datentyps des Rückgabewertes.

```
Function Func1( bParam1 As Bool ) As Numeric
Function Func2( bParam1 As Bool, nParam2 As Numeric )
Function Func3() As String
```

Bei einer internen Funktion folgt auf die Definition direkt der entsprechende Anweisungsteil. Das Ende der Funktion wird durch das Schlüsselwort *´End Function´* gekennzeichnet. Liefert die Funktion einen Rückgabewert, so wird dieser innerhalb des Anweisungsteiles dem Funktionsnamen zugewiesen. Ebenso kann einem Referenzparameter ein Ergebniswert zugewiesen werden.

```
Dim bReturn As Bool ´Variable für den Rückgabewert der Funktion
Dim nPersNr As Numeric ´Variable für den Rückgabewert der Funktion
Dim strName As String ´Variable für den Rückgabewert der Funktion

bReturn = Function BoolFunc( nPersNr, strName)

Function BoolFunc( nParam1 Ref Numeric, strParam2 As String ) As
Bool
Dim nLocalNr As Numeric ´Lokale Variable innerhalb der Funktion
   ... <Anweisungsteil>
   nParam1 = 123     ´Dem Referenzparameter wird ein Rückgabewert
                     ´zugewiesen
   BoolFunc = TRUE   ´Die Funktion selbst erhält einen Rückgabewert
End Function
´Inhalt der Variablen nach dem Funktionsaufruf
     => bReturn = True ; nPersNr = 123
```

Werden innerhalb des Anweisungsteils einer Funktion Variablen definiert (-> Funktionslokale Variable), so stehen diese außerhalb dieser Funktion nicht zur Verfügung.

Um auf die vom Druckmodul und der Applikation zur Verfügung gestellten Funktionen zugreifen zu können, müssen diese erst deklariert bzw. dem Kompilierer bekannt gegeben werden. Im Unterschied zu den internen Funktionen besitzen diese keinen Anweisungsteil und folgedessen auch kein *´End Function´*.

```
Declare external SetPos( x As Numeric, y As Numeric )
Declare external Now() As Numeric
Declare external PageBreak()
```

3.32 Seitenumbruch und Seitennummerierung

Mit der Funktion

PageBreak()

kann ein Seitenumbruch ausgelöst werden.
Innerhalb des Formulares kann die aktuelle Seitennummer über die Funktion

GetPageNumber() as Numeric

als Numeric-Wert ermittelt werden.
Wenn die Seitennummer als Text benötigt wird, kann dafür die Definition

PAGENUMBER

verwendet werden.

```
 Dim nSeitenNr As Numeric
 nSeitenNr = GetPageNumber() ´ Die Seitennummer als Numericwert
ermitteln
 TL( "Seite:" + PAGENUMBER ) ´ Die Seitennummer als String ermitteln
 PageBreak() ´ führt einen Seitenumbruch aus
```

3.33 Formular- und Applikationsname

Zur Ermittlung des Formularpfades, des Applikationsnamens und der Versionsnummer der Applikation stehen folgende Funktionen zur Verfügung:
Liefert den kompletten Pfad zur aktuellen Formulardatei.

GetFormFileName() as String

Ermittelt den Namen der Anwendung, die das Druckmodul verwendet.

GetProgramName() as String

Ermittelt die Version des Programms, welches das Druckmodul verwendet, und nicht die Version des Druckmoduls selbst!

GetProgramVersion() as String

```
Dim strPfad    As String
Dim strAplName As String
Dim strAplVers As String
strPfad = GetFormFileName()
strAplName = GetProgramName()
strAplVers = GetProgramVersion()
```

3.4 Die Formularbeschreibung

Zur Beschreibung einer Formulardatei wird das Schlüsselwort *Description* verwendet. Beschreibungsvariablen werden direkt in die ausführbare Datei übernommen und können vom Programm ausgelesen werden, ohne die Ausführung des Formulars zu starten.

Diese Beschreibungen werden zum Beispiel verwendet, um bei der Auswahl eines Formulars im jeweiligen Programm eine Beschreibung anzeigen zu können.

Dabei stehen folgende Beschreibungsvariablen zur Verfügung:

Form_Description	Beschreibung des Formulars. Mit dieser Beschreibung wird das Formular unter der Dialogseite ´Optionen´ des Druckdialoges aufgelistet.
Form_Code	Bereich, für den dieses Formular vorgesehen ist. Diese Eigenschaft wird vom aufrufenden Programm verwendet, um festzustellen, ob das Formular für den aktuellen Druckauftrag vorgesehen ist. Die Formcodes für die unterschiedlichen Berichte und Ausdrucke sind der jeweiligen programmbezogenen Dokumentation zu entnehmen.
Form_Version	Versionsnummer des Formulares in der Form “X.XX.XX.XXXX“ (Zusammensetzung entsprechend den Versionsnummern der Programme und DLL’s). Diese wird (in einigen Programmen...) in der Titelleiste des Druckdialoges angezeigt, wenn in der Registry im entspr. Programmast von HKEY_LOCAL_MACHINE unter *Info* der Wert *Lange Versionsnummer* auf 1 gesetzt ist.
Form_Target	Zielgerät, für das dieses Formular erstellt wurde. Diese Angabe ist optional. Im allgemeinen wird diese Eigenschaft angegeben, wenn das Formular

	speziell für den Druck in eine Datei (“File“) oder zur Steuerung eines OLE-Objektes ("COM") verwendet wird .
Page_Orientation	Seitenausrichtung des Formulars. Hierfür sind in der Datei DRUCKNT.INT bereits folgende Werte definiert: PAGE_PORTRAIT Hochformat der Seite PAGE_LANDSCAPE Querformat der Seite PAGE_STANDARD Die aktuelle Seitenorientierung des Druckers wird verwendet

Ein Beispiel für eine komplette Formularbeschreibung:

```
Description form_description As "Journal, Dateiexport"
Description form_code As "Journal"
Description Form_Version As "1.00.00.1026"
Description form_target As "File"
Description page_orientation As PAGE_LANDSCAPE
```

3.5 Druckbereiche eines Formulares

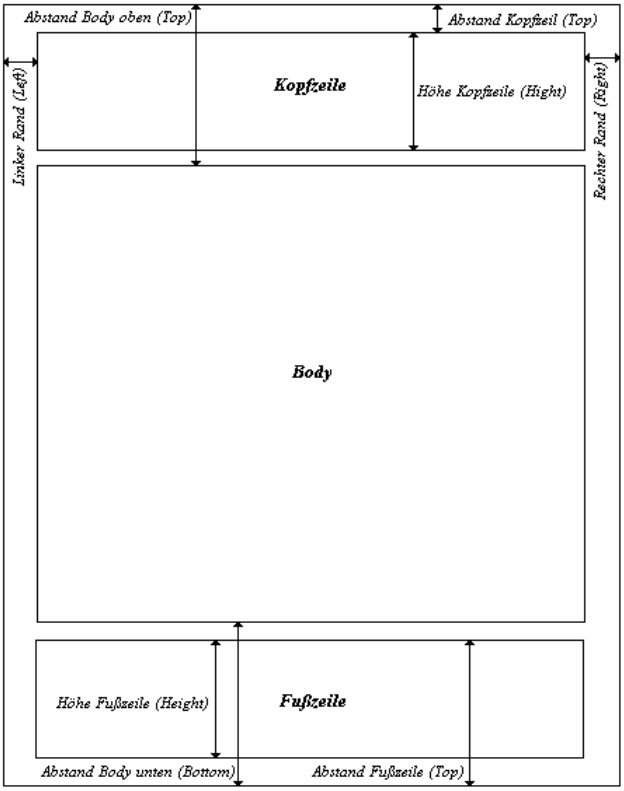
Ein Formular kann in folgende drei Bereiche unterteilt werden:

Kopfbereich
Rumpfbereich
Fußbereich

Die Größe der einzelnen Bereiche kann mit folgenden Funktionen eingestellt werden

SetHeaderMargin(left as Numeric, top as Numeric, right as numeric, height as Numeric)
SetBodyMargin(left as Numeric, top as Numeric, right as numeric, bottom as Numeric)
SetFooterMargin(left as Numeric, top as Numeric, right as numeric, height as Numeric)

Hierbei ist besonders darauf zu achten, dass durch das Bestimmen der einzelnen Bereiche keine Überschneidungen zustande kommen. Wenn z.B. der Abstand der Kopfzeile vom oberen Rand zusammen mit deren Höhe größer ist als der Abstand des Rumpfbereiches (Body) vom oberen Rand, so kann es durch die Überschneidung dazu führen, dass nicht alles, was im Formular ausgegeben wird, auch auf dem Ausdruck erscheint (Siehe Skizze).



Left

Abstand vom linken physikalischen Blattrand.

Right

Abstand vom rechten physikalischen Blattrand.

Top

Abstand vom oberen physikalischen Blattrand. bzw.
Abstand vom unteren physikalischen Blattrand

Bottom

Abstand vom unteren physikalischen Blattrand.

```
´links 10 mm, Abstand oben 9 mm, rechts 15 mm, Höhe Kopf 20 mm
SetHeaderMargin ( 100, 90, 150, 200 )
´links 10 mm, Abstand oben 30 mm, rechts 15 mm, Abstand unten 20 mm
SetBodyMargin( 100, 300, 150, 280 )
´links 10 mm, Abstand oben 20 mm, rechts 15 mm, Höhe Fuß 20 mm
SetFooterMargin ( 100, 200, 150, 100 )
```

Falls innerhalb des Skriptes die Höhe oder Breite des aktuellen Ausgabebereiches benötigt wird, um beispielsweise zu ermitteln, wieviel Platz noch zum Rand zur Verfügung steht, kann dies über die Funktionen

GetAreaLength() as Numeric
GetAreaWidth() as Numeric

abgefragt werden. Der Ausgabebereich ist entweder Kopfzeile, Fußzeile oder Seitenrumpf.
Diese Vorgehensweise ist dem Verwenden von festen Werten vorzuziehen, da dabei Änderungen der Randeinstellungen automatisch berücksichtigt werden.

Alle Positions-, Höhen- und Breitenangaben erfolgen grundsätzlich in 0.1mm. Die einzelnen Bereiche eines Formulares stellen jeweils ein eigenes Koordinatensystem dar, wobei die Ecke links oben durch die Koordinaten X = 0 und Y = 0 beschrieben werden. Die X-Koordinaten wachsen von links nach rechts, die Y-Koordinaten von oben nach unten.

```
Dim nAreaLenght As Numeric
Dim nAreaWidth  As Numeric

nAreaLenght = GetAreaLenght()
nAreaWidth = GetAreaWidth()
```

3.6 Behandlung von Events

Um innerhalb eines Druckformulars auf bestimmte Gegebenheiten reagieren zu können, wird vom Druckmodul in folgenden Fällen ein sogenanntes Ereignis (-> Event) ausgelöst:

Ereignis	Beschreibung
EVENT_START_PAGE	Dieses Ereignis wird ausgelöst, bevor der Druck einer neuen Seite beginnt.
EVENT_END_PAGE	Dieses Ereignis wird ausgelöst, nachdem die Bearbeitung einer Seite abgeschlossen wurde.
EVENT_PRINT_HEADER	Dieses Ereignis wird ausgelöst, wenn der Kopfbereich einer Seite gedruckt werden soll.
EVENT_PRINT_FOOTER	Dieses Ereignis wird ausgelöst, wenn der Fußbereich einer Seite gedruckt werden soll.
EVENT_BREAK_TABLE	Tritt ein, falls innerhalb einer Tabelle ein Seitenumbruch stattfindet.
EVENT_END_TABLE	Tritt ein, falls die letzte Zeile einer Tabelle gedruckt wurde.

Um auf die einzelnen Ereignisse zu reagieren, kann innerhalb eines Formulares für jedes Ereignis eine Funktion geschrieben werden, die dem entsprechenden Ereignis mit

SetEventHandler(eventID as Numeric, handlerFunction as Pointer)

zugeordnet wird. Wenn die Zuordnung einer Funktion zu einem Ereignis aufgehoben werden soll, kann dies über

DeleteEventHandler(eventID as Numeric)

erfolgen.

```
Function Kopfzeile()
      SetTabs( "L50 R-50" )
       TL( "\t" + strFirmenname + "\t" + FormatDate("%d.%m.%Y", Now()))
End Function
´ Setzen des Eventhandlers
SetEventHandler( EVENT_PRINT_HEADER, Kopfzeile )
```

3.7 Ausgabe von Linien

3.71 Linienfunktionen

Zur Ausgabe von Linien und Rechtecken stehen folgende Funktionen zur Verfügung:

Die Funktion

DrawLine(x as Numeric, y as Numeric)

zeichnet eine Linie mit dem aktuell eingestellten Stift von der aktuellen Position zur angegebenen Zielposition.

Die Funktion

DrawRect(x as Numeric, y as Numeric)

zeichnet ein Rechteck mit aktuellem Stift und aktuellem Pinsel. Der Punkt links oben ist die aktuelle Ausgabeposition, der rechte untere Punkt wird als Parameter angegeben.

Die Funktion

DrawLineExt(x1 as Numeric, y1 as Numeric, x2 as Numeric, y2 as Numeric)

zeichnet eine Linie mit dem aktuell eingestellten Stift von einer Start- zu einer Zielposition.

Die Funktion

DrawRectExt(x1 as Numeric, y1 as Numeric, x2 as Numeric, y2 as Numeric)

zeichnet ein Rechteck mit aktuellem Stift und aktuellem Pinsel. Die Ausmaße des Rechtecks sind spezifiziert durch den Punkt links oben und rechts unten.

Die Funktion

DrawHorzLine(bottom as Bool)

zeichnet eine horizontale Linie mit dem aktuell eingestellten Stift vom linken bis zum rechten Seitenrand.

Wenn der Parameter ´*True*´ ist, wird die Linie unterhalb der momentanen Textzeile gezeichnet, ansonsten oberhalb.

Diese Funktionen verwenden zur Ausgabe die jeweils aktuelle Linienart.

```
 ´ Zeichnet eine Linie von der aktuellen Position zur Position 100,
200
 DrawLine( 100, 200 )
 ´ Zeichnet ein Rechteck von der aktuellen Position zu der Position
1000,
 ´ 2000
 DrawRect( 1000, 2000 )
 ´ Zeichnet eine Linie von der Position 100,200 bis zur Position 500,
200
 DrawLineExt( 100, 200, 500, 200 )
 ´ Zeichnet eine ein Rechteck von der Position 100, 200 bis zur
Position
 ´ 1000, 2000
 DrawRectExt( 100, 200, 1000, 2000 )
 ´ Zeichnet eine Linie vom linken bis zum rechten Seitenrand
unterhalb der
 ´ momentanen Textzeile
 DrawHorzLine( True )
```

3.72 Linienarten

Entsprechend den Schriftarten besteht auch bei den Linienarten die Möglichkeit, über die Funktion

SelectPen(id as PenType)

die aktuelle Linienart zu setzen, wobei diese Funktion als Parameter einen Wert des Typs *Pentype* benötigt. Eine Variable des Typs *Pentype* wird über die Funktion

CreatePen(width as Numeric, type as Numeric, color as Numeric) as PenType

erzeugt. Über diese Funktion kann ein Stift mit den oben angegebenen Eigenschaften erzeugt werden. Der Vorteil einer Variablen des Types *Pentype* liegt darin, dass durch Ändern dieser Variablen alle Ausgaben, die mit diesem Stift vorgenommen werden, einheitlich angeglichen werden können. Bei Formularen, die umfangreiche Linienausgaben enthalten, sollte über den Mechanismus mit *CreatePen()* und *SelectPen()* vorgegangen werden, wobei auch hier zur einfacheren Anpassung die verwendeten Stifte am Anfang des Formulares definiert und erzeugt werden sollten.

Ein Stift wird durch die Linienbreite, die Linienart (durchgezogen, gepunktet oder gestrichelt) und die Farbe definiert. Für die Linienart und die Farbe können Konstanten, die in DruckNT.INT definiert sind, verwendet werden.

PEN_SOLID	durchgezogene Linie
PEN_DOT	gepunktete Linie
PEN_DASH	gestrichelte Linie
PEN_NULL	keine Linie (unsichtbar)

Eine besondere Eigenschaft eines Stiftes ist die Linienart ´PEN_NULL´, mit der ein unsichtbarer Stift erzeugt wird. Diese Eigenschaft wird vor allem im Zusammenhang mit Tabellen, die in einem gesonderten Kapitel beschrieben werden, benötigt. Bei den hier erwähnten Linienfunktionen kann mit einem unsichtbaren Stift ein Teil einer bereits vorhandenen Linie unsichtbar gemacht werden.

```
 ´ notwendige Variablen deklarieren
 Dim PEN_SOLID As PenType
```

```
Dim PEN_DOT   As PenType
Dim PEN_DASH  As PenType

´ Die verschiedenen Stifte erzeugen
PEN_SOLID = CreatePen( 10, PEN_SOLID, COLOR_BLACK )
PEN_DOT   = CreatePen( 5, PEN_DOT, COLOR_BLACK )
PEN_DASH  = CreatePen( 8, PEN_DASH, COLOR_BLACK )

´ den aktuellen Stift setzen
SelectPen( PEN_SOLID )
```

Wie bei den Schriftarten gibt es auch hier eine weitere Funktion, um die aktuelle Linienart zu wechseln.
Mit der Funktion

SetPen(width as Numeric, type as Numeric, color as Numeric)

können alle Eigenschaften des aktuellen Stiftes zusammengesetzt werden. Während die folgenden Funktionen jeweils nur eine Eigenschaft verändern, ohne die anderen zu beeinflussen.

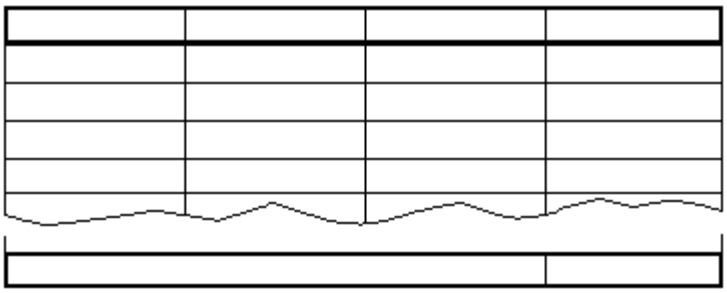
SetPenType(type as Numeric)
SetPenWidth(width as Numeric)
SetPenColor(color as Numeric)

3.8 Tabellen

3.81 Allgemein

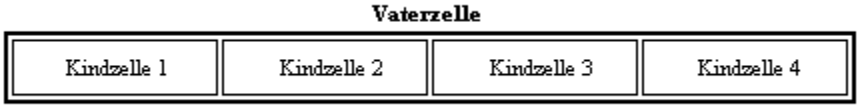
Die meisten Ausdrucke von Lexwareprogrammen stellen Tabellen dar. Prinzipiell könnte eine Tabelle durch die Ausgabe von Linien und Text mit entsprechenden Tabulatorpositionen ausgegeben werden. Wenn jedoch auf bestimmte Inhalte reagiert werden soll, ist dies eine sehr unflexible Methode. Aus diesem Grund gibt es die Möglichkeit, Tabellen zu definieren.

Nachfolgend wird Schritt für Schritt der Aufbau einer einfachen Tabelle, die aus einer Kopfzeile, den eigentlichen Tabellenzeilen und einer Fußzeile besteht, beschrieben. (Siehe Abbildung)



3.82 Aufbau einer Tabelle aus Zellen

Tabellen werden bei DruckNT aus hierarchisch angeordneten Zellen aufgebaut. Eine Zeile setzt sich aus einer Vaterzelle, die die gesamte Zeile repräsentiert und den einzelnen Kindzellen, die die einzelnen Spalten repräsentieren,zusammen.



Über die Eigenschaften der Vaterzelle kann bestimmt werden, ob dessen Kindzellen horizontal (wie im obigen Beispiel) oder vertikal (dies ist dann sinnvoll, wenn sich zum Beispiel der Kopf einer Tabelle aus mehreren Zeilen zusammensetzt) angeordnet werden.

Zum Definieren von Zellen wird der Datentyp *CellType* verwendet. Über die Funktion

> *CreateCell(width as Numeric, height as Numeric, flags as Numeric) as CellType*

können Zellen erzeugt werden. Über die Parameter ´width´ und ´height´ werden die minimale Breite und Höhe der Zelle eingestellt. Der Parameter ´flags´ bestimmt die Eigenschaften der Zelle, er kann aus mehreren Konstanten zusammengesetzt werden. Dazu werden die Konstanten mit einer Addition verknüpft.

Folgende Konstanten sind für diesen Parameter erlaubt:

CELL_TYPE_DYNHORZ	Die Zelle hat in horizontaler Richtung eine flexible Größe
CELL_TYPE_DYNVERT	Die Zelle hat in vertikaler Richtung eine flexible Größe

CELL_TYPE_HORZ	Die Nachfolger (Kindzellen) dieser Zelle sind horizontal angeordnet
CELL_TYPE_VERT	Die Nachfolger (Kindzellen) der Zelle sind vertikal angeordnet
CELL_TYPE_CLIP	Die Ausgabe des Inhalts wird auf die Ausmaße der Zelle beschränkt. Der Rest wird abgeschnitten
CELL_LINE_SEPARATOR	Nach jeder Textzeile innerhalb einer Zelle wird eine Trennlinie gezeichnet. Außerdem werden die Abstände von den Zellenrändern eingehalten.
CELL_WORD_WRAP	Ist nur erlaubt, falls die Zelle eine feste Breite hat. Der Text wird am rechten Rand umgebrochen. Die Trennstelle ist immer ein Leerzeichen. Das heißt, es werden nur ganze Wörter umgebrochen. Ein automatischer Zeilenumbruch funktioniert nur bei linksbündig ausgerichteten Zellen.

Die Funktion

CreateCellEx(parent as CellType, width as Numeric, height as Numeric, flags as Numeric, text asString, font as FontType, align as Numeric, color as Numeric, pLeft as PenType, pTop as PenType, pRight as PenType, pBottom as PenType, pSep as PenType, bLeft as Numeric, bTop as Numeric, bRight as Numeric, bBottom as Numeric)
as CellType

erzeugt ebenfalls eine neue Zelle, wobei hier schon alle möglichen Attribute, die die Eigenschaften und das Aussehen der Zelle bestimmen, direkt gesetzt werden können. Da diese Definition einer Zelle nicht gerade sehr übersichtlich ist sollte sie bei umfangreicheren Tabellen vermieden werden.

Mit der Funktion

DuplicateCell(id as CellType) as Numeric

ist es möglich, bereits definierte Zellen zu kopieren. Auf die letzten beiden Funktionen wird hier nicht weiter eingegangen.

Als erstes werden für die einzelnen Zeilen die Zellen definiert

```
' Definition der notwendigen Zellen
' Kopfzeile
Dim cKopf     As CellType
' Kopfspalten
Dim cKSpalte1 As CellType
Dim cKSpalte2 As CellType
Dim cKSpalte3 As CellType
Dim cKSpalte4 As CellType
' Tabellenzeile
Dim cZeile    As CellType
' Tabellenspalten
Dim cSpalte1  As CellType
Dim cSpalte2  As CellType
Dim cSpalte3  As CellType
Dim cSpalte4  As CellType
' Fußzeile
Dim cFuss     As CellType
' Fußspalten
Dim cFSpalte1 As CellType
Dim cFSpalte2 As CellType
' Vaterzelle der Kopfzeile
cKopf = CreateCell(0, 0, CELL_TYPE_DYNHORZ + CELL_TYPE_DYNVERT +
              CELL_TYPE_HORZ)
' Kindzellen der Kopfzeile
cKSpalte1 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
cKSpalte2 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
cKSpalte3 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
cKSpalte4 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
' Vaterzelle der Tabellenzeile
cZeile = CreateCell(0, 0, CELL_TYPE_DYNHORZ + CELL_TYPE_DYNVERT +
               CELL_TYPE_HORZ)
' Kindzellen der Tabellenzeile
cSpalte1 = CreateCell(300, 0, CELL_TYPE_DYNVERT + CELL_WORD_WRAP)
cSpalte2 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
cSpalte3 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
cSpalte4 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
' Vaterzelle der Fußzeile
cFuss = CreateCell(0, 0, CELL_TYPE_DYNHORZ + CELL_TYPE_DYNVERT +
              CELL_TYPE_HORZ)
' Kindzellen der Fußzeile
cFSpalte1 = CreateCell(300 * 3, 0, CELL_TYPE_DYNVERT)
cFSpalte2 = CreateCell(300, 0, CELL_TYPE_DYNVERT)
```

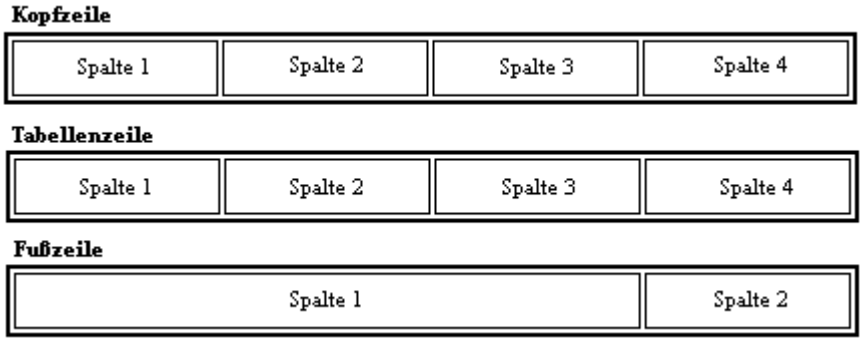
Als nächster Schritt wird den Kindzellen die Vaterzelle zugewiesen. Allerdings kann jede Zelle nur einen Vater besitzen. Über die Funktion

SetCellParent(id as CellType, parent as CellType)

wird die Vaterzelle für eine Zelle bestimmt.

```
´Zuweisungen für die Kopfzeile
SetCellParent( cKSpalte1, cKopf )
SetCellParent( cKSpalte2, cKopf )
SetCellParent( cKSpalte3, cKopf )
SetCellParent( cKSpalte4, cKopf )
´Zuweisungen für die Tabellenzeile
SetCellParent( cSpalte1, cZeile )
SetCellParent( cSpalte2, cZeile )
SetCellParent( cSpalte3, cZeile )
SetCellParent( cSpalte4, cZeile )
´Zuweisungen für die Fusszeile
SetCellParent( cFSpalte1, cFuss )
SetCellParent( cFSpalte2, cFuss )
```

Die Tabelle hat nun folgendes Aussehen:



Damit der Kopf der Tabelle jeweils zu Beginn einer neuen Tabelle und nach einem Seitenumbruch innerhalb einer Tabelle automatisch ausgegeben wird, muss dieser noch über die Funktion

SetHeaderCell(id as CellType *)*

als aktueller Kopf gesetzt werden. Das gleiche gilt für die Fußzeile. Der Fuß einer Tabelle wird jeweils am Ende einer Tabelle und vor einem

Seitenumbruch innerhalb einer Tabelle automatisch ausgegeben. Die entsprechende Funktion hierfür heißt

SetFooterCell(id as CellType)

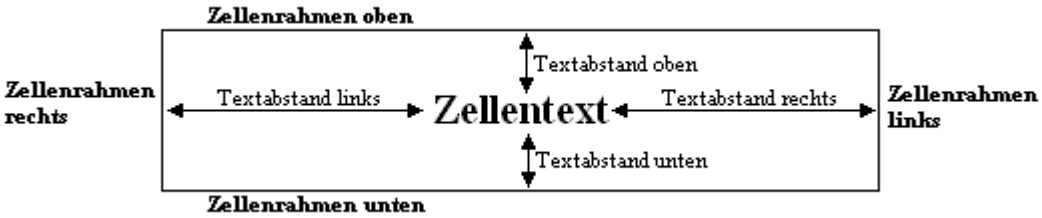
```
SetHeaderCell( cKopf )
SetFooterCell( cFuss )
```

Der Aufbau der Tabelle ist ab hier abgeschlossen und mit dem Aufruf der Funktion ´DrawCell(cZeile, False) kann jetzt zum Testen eine leere Tabelle ausgegeben werden.

3.83 Eigenschaften einer Zelle

Eine Zelle verfügt noch über zahlreiche Eigenschaften mehr, die das Erscheinungsbild bestimmen. Die Eigenschaften einer mit *CreateCell()* erzeugten Zelle können alle über entsprechende Funktionen gesetzt werden. Die Eigenschaften können auch direkt beim Erzeugen der Zelle mit der Funktion *CreateCellEx()* bestimmt werden. Zur nachträglichen Veränderung einer oder mehrerer Eigenschaften wird dann die jeweilige Funktion verwendet.

Folgende Eigenschaften einer Zelle können definiert werden



Eigenschaft	Beschreibung	Funktionen
Vaterzelle	Definiert die Vaterzelle.	SetCellParent() GetCellParent()
Breite u. Höhe	Die Breite und Höhe der Zelle wird beim Erzeugen in 0.1mm angegeben. Bei Zellen, die eine dynamische Höhe oder Breite	CreateCell() GetCellWidth() GetCellHeight()

	haben (siehe auch unter der Eigenschaft Art), kann jeweils der Wert 0 angegeben werden.	
Art	Über die Art kann festgelegt werden, ob die Kindzellen der Zelle horizontal oder vertikal angeordnet werden, ob die Höhe oder Breite der Zelle dynamisch (also abhängig von den Kindzellen oder vom enthaltenen Text) ist, ob der auszugebende Text abgeschnitten werden soll, ob bei mehrzeiligen Zellen eine Trennlinie zwischen den Zeilen gezeichnet werden soll und ob ein automatischer Zeilenumbruch bei Ausgabe von längeren Texten erfolgen soll.	CreateCell() SetCellClip()
Text	Auszugebender Text.	SetCellText()
Schriftart	Schriftart, in der der Text ausgegeben werden soll. Bei Zellen mit dynamischer Höhe ergibt sich diese aus der Höhe der gesetzten Schriftart und den eingestellten Texträndern.	SetCellFont() SelectCellFont()
Ausrichtung	Ausrichtung des auszugebenden Textes: rechtsbündig, zentriert oder linksbündig.	SetCellAlign()
Hintergrundfarbe	Hintergrundfarbe der Zelle.	SetCellBackground()
Rahmenlinien	Die Linienart des Rahmens separat für links, oben, rechts, unten und die Zeilentrennlinie einstellbar.	SetCellBorder() SelectCellBorder()
Texträander	Abstand der Zellränder vom auszugebenden Text separat für links, oben, rechts und unten einstellbar.	SetCellTextBorder()

Einige dieser Eigenschaften werden nachfolgend noch näher beschrieben.

3.84 Aussehen der Tabelle verändern

Als nächster Schritt wird das Erscheinungsbild der Tabelle verändert. Die Zellen der Kopfzeile sollen eine Hintergrundfarbe und stärkere Rahmenlinien

erhalten. In der Tabelle selbst sollen die horizontalen Linien nicht zu sehen sein, und der Tabellenfuß soll ebenfalls stärkere Rahmenlinien erhalten.

Begonnen wird mit dem Tabellenkopf und dem Setzen der Hintergrundfarbe. Hierfür wird die Funktion

SetCellBackground(id as CellType, color as Numeric)

benötigt. Für den Parameter ´color´ stehen folgende Farbkonstanten zur Verfügung

COLOR_TRANSPARENT
COLOR_BLACK
COLOR_WHITE
COLOR_RED
COLOR_GREEN
COLOR_BLUE
COLOR_LIGHTGREY

Die Farbkonstanten können dazu verwendet werden, um neue Farben zu mischen. Dazu kann man einfach die Anteile rot, grün und blau an der gewünschten Farbe mit Hilfe der Konstanten aufaddieren. Die Farbe COLOR_TRANSPARENT wird verwendet, um etwas unsichtbar zu machen.

Um die Rahmenlinien der Zellen zu beeinflussen, wird die Funktion

SetCellBorder(id as CellType, which as Numeric, width as Numeric, type as Numeric,color as Numeric)

benötigt. Für den Parameter ´which´ stehen ebenfalls Konstanten zur Verfügung

CELL_BORDER_TOP	oberer Rand
CELL_BORDER_BOTTOM	unterer Rand
CELL_BORDER_LEFT	linker Rand
CELL_BORDER_RIGHT	rechter Rand
CELL_BORDER_ALL	alle Ränder sind gemeint

Mit dem Parameter ´width´ wird die Linienstärke festgelegt, wobei 0 eine unsichtbare darstellt. Über den Parameter ´type´ wird eine Stiftkonstante festgelegt. Diese wurden schon weiter oben unter Linienarten besprochen.

```
´ setzen der Hintergrundfarbe für die Kopfzeile
 SetCellBackground( cKopf, COLOR_GREEN + COLOR_RED )´Gelber
Hintergrund
 ´ setzen der Rahmeneinstellungen für die Kopfzeile
 SetCellBorder( cKopf, CELL_BORDER_ALL, 3, PEN_SOLID, COLOR_BLACK )
 SetCellBorder( cKSpalte1, CELL_BORDER_ALL, 3, PEN_SOLID, COLOR_BLACK
)
 SetCellBorder( cKSpalte2, CELL_BORDER_ALL, 3, PEN_SOLID, COLOR_BLACK
)
 SetCellBorder( cKSpalte3, CELL_BORDER_ALL, 3, PEN_SOLID, COLOR_BLACK
)
 SetCellBorder( cKSpalte4, CELL_BORDER_ALL, 3, PEN_SOLID, COLOR_BLACK
)
´ setzen der Rahmeneinstellungen für den Kopf
 SetCellBorder( cZeile, CELL_BORDER_ALL, 0, PEN_NULL, COLOR_BLACK )
 SetCellBorder( cSpalte1, CELL_BORDER_TOP + CELL_BORDER_TOP, 3,
PEN_NULL,
                COLOR_BLACK )
 SetCellBorder( cSpalte2, CELL_BORDER_TOP + CELL_BORDER_TOP, 3,
PEN_NULL,
                COLOR_BLACK )
 SetCellBorder( cSpalte3, CELL_BORDER_TOP + CELL_BORDER_TOP, 3,
PEN_NULL,
                COLOR_BLACK )
 SetCellBorder( cSpalte4, CELL_BORDER_TOP + CELL_BORDER_TOP, 3,
PEN_NULL,
                COLOR_BLACK )
```

```
 ´ setzen der Rahmeneinstellungen für die Fußzeile
 SetCellBorder( cFuss, CELL_BORDER_ALL, 3, PEN_SOLID, COLOR_BLACK )
 SetCellBorder( cFSpalte1, CELL_BORDER_TOP + CELL_BORDER_BOTTOM +
                CELL_BORDER_LEFT, 3, PEN_SOLID, COLOR_BLACK )
 SetCellBorder( cFSpalte2, CELL_BORDER_TOP + CELL_BORDER_BOTTOM +
                CELL_BORDER_RIGHT, 3, PEN_SOLID, COLOR_BLACK )
```

Nun besitzt die Tabelle das gewünschte Aussehen. Um dies zu überprüfen, können wir die Tabelle wieder über den Funktionsaufruf ´DrawCell(cZeile, False) ausgeben.

3.85 Tabellen mit Text füllen

Nachdem nun die Tabelle definiert wurde und das Aussehen der Zellen festgelegt ist, wird sie mit Text gefüllt.

Der Text einer Zelle kann über die Funktion

SetCellText(id as CellType, text as String)

gesetzt werden. Der auszugebende Text kann hierbei ebenso wie bei der normalen Textausgabe über alle verfügbaren Operatoren oder Funktionen zusammengesetzt werden. Die Ausrichtung des Textes innerhalb einer Zelle wird mit der Funktion

SetCellAlign(id as CellType, align as Numeric)

bestimmt. Hierzu stehen folgende Konstanten zur Verfügung:

CELL_TEXT_LEFT
CELL_TEXT_RIGHT
CELL_TEXT_CENTER

Die Vorgehensweise, um die Schriftart in den Zellen festzulegen, ist fast die gleiche, wie unter dem Abschnitt Schriftarten. Dafür stehen zwei Funktionen zur Verfügung. Einmal die Funktion

SetCellFont(id as CellType, name as String, size as Numeric, type as Numeric, color as Numeric)

mit der wie bei *´SetFont()´* eine Schriftart für eine Zelle deklariert wird und die Funktion

SelectCellFont(id as CellType, font as FontType)

Wobei die Funktion *´SelectCellFont()´* die bessere von beiden ist, da mit ihr auf Schriftarten zugegriffen werden kann, die mit *´CreateFont´* deklariert wurden. Das heißt, man muss alle benötigten Schriftarten nur einmal am Anfang des Formulars deklarieren und kann dann im ganzen Formular auf diese Schriftart zugreifen, unabhängig davon, ob diese für die Ausgabe in einer Zelle oder mit *´DrawText()´* verwendet wird.

Bei der Ausgabe von mehreren Tabellenzeilen nacheinander ist zu beachten, dass eine Zelle ihren gesetzten Zelltext so lange beibehält, bis ein neuer Text gesetzt wird. Enthält eine Tabelle Spalten, die nicht in jeder Zeile einen Text enthalten, so muss darauf geachtet werden, dass diese Zellen vor der Ausgabe mit einem Leertext initialisiert werden, da sie ansonsten noch den Text der zuvor ausgegebenen Zeile enthalten.

```
´ setzen der Textausrichtung für die Kopfzeile
SetCellAlign( cKSpalte1, CELL_TEXT_CENTER )
SetCellAlign( cKSpalte2, CELL_TEXT_CENTER )
SetCellAlign( cKSpalte3, CELL_TEXT_CENTER )
SetCellAlign( cKSpalte4, CELL_TEXT_CENTER )
´ setzen der Textausrichtung für die Tabellenzeile
SetCellAlign( cSpalte1, CELL_TEXT_LEFT )
SetCellAlign( cSpalte2, CELL_TEXT_CENTER )
SetCellAlign( cSpalte3, CELL_TEXT_RIGHT )
SetCellAlign( cSpalte4, CELL_TEXT_RIGHT )
´ setzen der Textausrichtung für die Kopfzeile
SetCellAlign( cFSpalte1, CELL_TEXT_LEFT )
SetCellAlign( cFSpalte2, CELL_TEXT_RIGHT )
```

```
´ setzen der Schriftart für die Kopfzeile
SelectCellFont( cKSpalte1, ARIAL_FETT_10 )
SelectCellFont( cKSpalte2, ARIAL_FETT_10 )
SelectCellFont( cKSpalte3, ARIAL_FETT_10 )
SelectCellFont( cKSpalte4, ARIAL_FETT_10 )
´ setzen der Schriftart für die Tabellenzeile
SelectCellFont( cSpalte1, ARIAL_8 )
SelectCellFont( cSpalte2, ARIAL_8 )
SelectCellFont( cSpalte3, ARIAL_8 )
SelectCellFont( cSpalte4, ARIAL_8 )
´ setzen der Schriftart für die Fußzeile
SelectCellFont( cFSpalte1, ARIAL_FETT_10 )
SelectCellFont( cFSpalte2, ARIAL_FETT_8 )
´ setzen der Tabellenüberschrift in der Kopfzeile
SetCellText( cKSpalte1, "Artikeltext" )
SetCellText( cKSpalte2, "Menge" )
SetCellText( cKSpalte3, "Einzelpreis" )
SetCellText( cKSpalte4, "Gesamtpreis" )
´ setzen der Beschriftung in der Fußzeile
SetCellText( cFSpalte1, "Summe" )
```

3.86 Ausgabe einer Tabelle

Eine Tabellenzeile wird über die Funktion

DrawCell(id as CellType, continue as Bool)

ausgegeben, wobei diese Funktion grundsätzlich die übergebene Zelle und sämtliche Kindzellen dieser ausgibt. Bei der ersten Ausgabe einer Zelle wird eine evtl. definierte Kopfzeile automatisch ausgegeben. Die Ausgabe erfolgt horizontal am linken Rand des aktuellen Druckbereichs, vertikal an der aktuellen Ausgabeposition. Vor dem Ausgeben der Zelle prüft das Druckmodul, ob der im aktuellen Druckbereich noch zur Verfügung stehende Platz zur Ausgabe der angegebenen Zelle und eine evtl. definierte Fußzeile noch ausreicht. Ist dies nicht der Fall, so wird die definierte Fußzeile ausgegeben, ein Seitenvorschub ausgelöst und auf der neuen Seite zuerst die definierte Kopfzeile und dann die auszugebende Zelle gedruckt. Über einen Parameter kann mit der Funktion *DrawCell()* angegeben werden, ob es sich bei der auszugebenden Zelle um die letzte Zelle der Tabelle handelt. Ist dies der Fall, so wird nach dem Drucken der Zelle automatisch die definierte Fußzeile ausgegeben.

```
´ in Tabelle ausgeben
Dim i     As Numeric
Dim nSumme As Numeric
i = 1
While i <= 10 Do
       SetCellText( cSpalte1, "Artikeltext" + FormatNumeric("%.0f", i))
      SetCellText( cSpalte2, FormatNumeric("%.0f", i))
      SetCellText( cSpalte3, FormatNumeric("%.0f", i * i))
      SetCellText( cSpalte4, FormatNumeric("%.0f", i * i * i))

      nSumme =  nSumme + (i * i * i)
      i = i + 1

      If i < 10 Then
            DrawCell( cZeile, True)
      Else
            SetCellText( cSpalte1, FormatNumeric("%.0f", nSumme))
            DrawCell( cZeile, False)
      End If

End While
```

3.87 Kopfzeile, Fußzeile, Zwischensummen und Übertrag im Detail

Bei der Verwendung von Kopf- und Fußzeilen gibt es vor allem beim Einsatz von Zwischensummen und Übertrag einige Dinge zu beachten. Da vor dem Ausgeben einer Zeile noch nicht feststeht, ob diese Zeile noch auf die aktuelle Seite passt, muss vor der Ausgabe der Zeile eine Überprüfung des noch zur Verfügung stehenden Platzes gemacht werden. Außerdem dürfen Summenwerte, die in der Summen- und Übertragzeile ausgegeben werden, nicht vor der Ausgabe der Zeile berechnet werden. Auch sollte keine Fußzeile mit *´SetFooterCell()´* deklariert werden.
Als erstes muss man wissen, wieviel Platz mindestens noch benötigt wird. Das ist meistens eine Tabellenzeile plus der Zeile für die Zwischensumme. Um herauszufinden, wieviel Platz die beiden Zeilen benötigen, wird deren Höhe mit der Funktion *´GetCellHeight´* ermittelt und zu der momentan aktuellen Ausgabeposition, welche die Funktion *´GetPosY()´* liefert, hinzuaddiert. Das Ergebnis wird dann mit der zur Verfügung stehenden Länge des aktuellen Ausgabebereichs verglichen. Die Länge des Ausgabebereichs wird mit der Funktion *´GetAreaLength´* ermittelt.

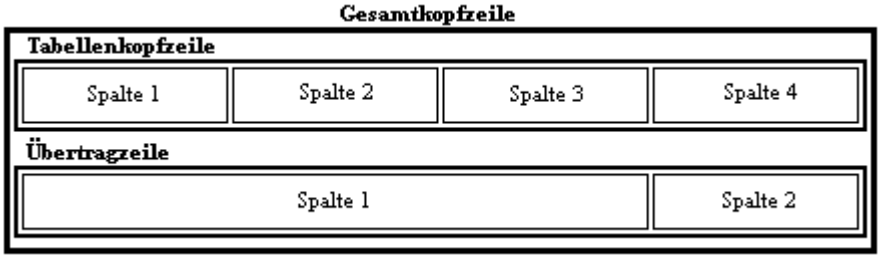
```
Function PlatzPruefen()
   Dim nZeilenHoehe As Numeric
   nZeilenHoehe = GetCellHeight(cZeile) + GetCellHeight(cFuss)

   ´ prüfen ob der Platz noch ausreicht
   If GetAreaLength() < GetPosY()+ nZeilenHoehe Then
     ´ wenn der Platz nicht mehr ausreicht
     SetCellText(cFSpalte1, "Zwischensumme")
     SetCellText(cFSpalte2, FormatNumeric("%.2f", nSumme))
     DrawCell(cFuss, True) ´ Zwischensumme ausgeben

     PageBreak() ´ Seitenumbruch auslösen

     SetCellText(cUSpalte2, FormatNumeric("%.2f", nSumme))
     DrawCell(cUebertrag, True) ´ Übertrag ausgeben
  End If
  nSumme =  nSumme + nBetrag
End Function
```

Um eine Übertragzeile einzubinden, gibt es mehrere Möglichkeiten.
Man deklariert eine eigene Übertragzeile (wie im Beispiel oben angedeutet) und benutzt für die Zwischensumme und Gesamtsumme die gleiche Zelle. In

diesem Fall muss dann nur der Zellentext (´Zwischen- , Gesamtsumme´) in der Summenzeile an der entsprechenden Stelle im Formular richtig gesetzt werden.

Wenn die Summenzeilen und die Übertragzeile das gleiche Aussehen haben, kann hierfür auch ein und dieselbe Zeile verwendet werden. Dann gilt das gleiche wie oben, dass an der entsprechenden Stelle im Formular der Zellentext richtig gesetzt werden muss.

Die eleganteste Lösung ist die, dass der Tabellenkopf aus 2 Zeilen zusammengesetzt wird. Da auf der ersten Seite im Normalfall keine Übertragzeile erwünscht ist, muss zunächst die eigentliche Kopfzeile als Tabellenkopf gesetzt werden. Nach der Ausgabe der ersten Zeile, mit der automatisch der Kopf ausgegeben wird, muss dann die Gesamtkopfzeile gesetzt werden. Danach erfolgt bei jedem Seitenumbruch (´*PageBreak()*´) automatisch die Ausgabe der Gesamtkopfzeile mit der Übertragzeile.



Die Zellen ´Tabellenkopfzeile´ und ´Übertragzeile´ sind Kindzellen der Zelle ´Gesamtkopfzeile´, wobei diese als Zelle mit vertikal angeordneten Kindzellen definiert werden muss.
Die beiden Kindzellen, ´Tabellenkopfzeile´ und ´Übertragzeile´, enthalten wiederum selbst Kindzellen, welche dann die einzelnen Spalten darstellen.

3.9 Ausgabe von Grafik

Zur Ausgabe von Grafiken steht die Funktion

DrawBitmap(x as Numeric, y as Numeric, width as Numeric, height as Numeric, file as String)

zur Verfügung, mit der Bitmapdateien ausgegeben werden können. Die Position, Höhe und Breite der Bitmap können angegeben werden. Die Datei kann entweder im gleichen Verzeichnis wie die Formulardatei liegen, oder kann mit kompletter Pfadangabe aus einem beliebigen Verzeichnis geladen werden. Diese Funktion hat keine Auswirkung auf die aktuelle Ausgabeposition.

```
´ Bitmap ausgeben
DrawBitmap( 100, 200, 400, 400, "C:\\Bitmaps\\Test.bmp")
```

Bei der Angabe eines Verzeichnispfades ist darauf zu achten, dass für Unterverzeichnisse immer ein doppeltes Backslash angegeben wird. Dies kommt daher, dass nach einem Backslash normalerweise ein Steuerzeichen folgt wie z.B. ´\n´ für einen Zeilenumbruch.

Zusätzlich stehen zur Ausgabe von Grafiken in anderen Dateiformaten die Befehle

DrawJPG (Grafikdateien im .JPG- oder .JPEG-Format),

DrawGIF (Grafikdateien im .GIF-Format) und

LXPlayMetafile (Abspielen von Enhanced Metafifes, d.h. .emf-Dateien)

zur Verfügung. Diese Befehle benutzen die gleiche Parameterliste wie der DrawBitmap-Befehl.

3.91 Zugriff auf Programmdaten

Bei den in diesem Kapitel beschriebenen Funktionen handelt es sich um Schnittstellenfunktionen, die von den einzelnen Applikationen zur Verfügung gestellt werden. Da diese Funktionen nicht Bestandteil des Druckmoduls sind, können Formulare, die diese Funktionen enthalten, nicht über die Seitenvorschaufunktion des Formulareditors getestet werden.

Grundsätzlich muss bei einem Formular sichergestellt werden, dass es nur für den vorgesehenen Druckauftrag der Zielapplikation verwendet wird. Durch Festlegen des Formcodes (siehe auch unter ‚Die Formularbeschreibung') wird definiert, für welchen Druckauftrag das Formular geeignet ist.

Allgemeine Datenfelder, wie Firmenangaben oder Angaben, die im aktuellen Druckauftrag übergreifend benötigt werden, stehen grundsätzlich an jeder beliebigen Stelle des Formulares zur Verfügung. Bei datensatzabhängigen Werten, die von einem bestimmten Mitarbeiter, einer Abrechnung, eines Buchungssatzes, eines Auftrages oder Vergleichbarem abhängig sind, muss zunächst der entsprechende Datensatz gelesen werden.

Zum Lesen von Datensätzen und Datenfeldern werden von der jeweiligen Applikation Bereichskennungen und Variablennummern vergeben. Diese Kennungen und Nummern sind der jeweiligen Programmdokumentation zu entnehmen. In einigen Fällen sind die Kennungen und Nummern in einer applikationsspezifischen Interfacedatei als Konstanten definiert, um das Erstellen von Formularen zu vereinfachen und deren Lesbarkeit zu verbessern.

3.92 Lesen von Datensätzen

In Formularen, in denen mehrere Datensätze z.B. in einer Tabelle oder in Form von mehreren Seiten je Datensatz ausgegeben werden, müssen die Datensätze eingelesen werden. Dazu stellen die Lexware Applikationen folgende Funktionen zur Verfügung:

FirstRecord()
NextRecord()

Mit der Funktion FirstRecord() wird der erste spezifizierte Datensatz eingelesen, mit der Funktion NextRecord() können in einer Schleife alle weiteren Datensätze eingelesen werden. Welche Datensätze für den jeweiligen Ausdruck verfügbar sind, ist im Allgemeinen von der Auswahl innerhalb der jeweiligen Applikation abhängig (Zeitraum, andere Angaben Von-Bis, ...).

3.93 Lesen von Datenfeldern

Zum Lesen von bestimmten Datenfeldern stehen folgende Funktionen zur Verfügung

Funktion	Datentyp	Beschreibung
Get()	String	Liefert das angeforderte Datenfeld vom Typ String
Is()	Bool	Liefert das angeforderte Datenfeld vom Typ Bool
GetTime()	Numeric (Uhrzeit)	Liefert die angeforderte Uhrzeit als Numeric
GetDate()	Numeric (Datum)	Liefert das angeforderte Datum als Numeric
GetNumeric()	Numeric	Liefert das angeforderte Datenfeld vom Typ Numeric
GetCurrencyFormat()	String	Liefert den angeforderten Betrag als String in der angegebenen Währung mit der angegebenen Formatierung
GetCurrencyValue()	Numeric	Liefert den angeforderten Betrag als Numeric in der angegebenen Währung
GetCurrencyFormatValue()	String, ref Numeric	Liefert den angeforderten Betrag als String in der angegebenen Währung mit der angegebenen Formatierung und in einem Referenzparameter als Numeric
GetCurrencySymbol()	String	Liefert das Währungssymbol des angeforderten Betrags als String
FormatCurrency()	String	Formatiert den übergebenen Betrag als String mit der angegebenen Formatierung

Bei den ´Currency´ - Funktionen besteht die Möglichkeit, den Wert in einer bestimmten Währung (CC_DEM, CC_EUR, CC_ATS), in der Originalwährung (CC_ORG) oder in der Firmenwährung, die zuvor in eine Numeric-Variable eingelesen wird, anzufordern. Die Formatierungsfunktionen können das Währungssymbol rechts oder links vom Betrag angeben oder unterdrücken. Ebenso besteht die Möglichkeit, dass Beträge mit dem Wert 0.00 unterdrückt werden.

3. 10 Ausdrücke

Ein Ausdruck ist eine Formel zur Berechnung eines Wertes. In dieser Formel werden Funktionen (eigentlich deren Rückgabewerte), Konstanten und Variablen mit Hilfe von Operatoren verknüpft. Als Resultat bleibt ein Wert, der einen bestimmten Datentyp hat. Für die Steuerung der Bearbeitungsreihenfolge von Ausdrücken können Klammern gesetzt werden.

Beispiel
(n + 1) * 10

n Zuvor definierte Variable as Numeric (Zahl)
+ 1 Auf den Wert der Variable n wird +1 aufaddiert
* 10 Die Summe n + 1 wird mit dem Faktor 10 multipliziert

Beispiel

```
Dim n asNumeric = 3
a=(n + 1) * 10
```

Die Variable n beinhaltet nach der Berechnung immer noch den Wert 3. Die Variable a beinhaltet das Ergebnis 40

4.0 Lernbeispiele

Im Verzeichnis **KursFormulare**
finden Sie 16 Lernbeispiele für die Erstellung
von Formulardateien (Kurs01.lsf bis Kurs16.lsf)
und die Dateien DruckNT.int, DruckExt.lsf,
LexVM.int (Schnittstellendateien zum Druckmodul)

Anhand der Beispiele können Sie nach und nach die Funktionen des Druckmoduls kennenlernen.

1. Starten Sie den Formular Editor.

2. Öffnen Sie im Verzeichnis **KursFormulare** ein Formularbeispiel.

3. Im geöffneten Fenster finden Sie die Beschreibung des Formulars. Das Formular ist jedoch noch nicht als Projekt geladen (Anzeige: "Kein Formular - DruckNT IDE" in der Fensterleiste). Beachten Sie bitte, dass das Formular zuerst als Projekt geladen werden muss.

4. Laden Sie das gewünschte Formular als aktives Projekt in den Formulareditor unter dem Menüpunkt "Projekt/Formular" und übersetzen Sie es mit der F7 - Taste (oder über den Menüpunkt "Projekt/Kompilieren").

   WICHTIG: Erst durch Kompilieren wird aus der Scriptdatei (LSF-Datei) die Formulardatei (gleichnamige VMB-Datei) erstellt. Das Ergebnis können Sie dann in der Seitenvorschau mit F5 ansehen.

5. Nach jeder Änderung muss das Projekt mit der F7 - Taste (oder Menüpunkt "Projekt/Kompilieren") neu übersetzt werden, bevor die Änderungen wirksam und über die Seitenvorschau mit F5 sichtbar werden.

TIPP: Setzen Sie im Beispielformular mit der Maus den Cursor auf einen Programmbefehl (z.B. "SetFont") und starten die Onlinehilfe mit der F1-Taste. Dadurch erhalten Sie direkte Hilfe zu dem entsprechenden Befehl. Weitergehende Hilfe bietet Ihnen auch die Programmhilfe unter dem Menüpunkt "**?**".

Die Beispieldateien sollen Ihnen beim Erstellen von Formulardateien behilflich sein.

Kurs01.lsf - Einfache Textausgabe
Kurs02.lsf - Einfache Textausgabe mit unterschiedlichen Schriften
Kurs03.lsf - Einfache Textausgabe mit vorgegebenem Rand
Kurs04.lsf - Einfache Textausgabe mit vorgegebenem Rand und Tabulatoren
Kurs05.lsf - Einfache Textausgabe mit vorgegebenem Rand, Tabulatoren, Linien
Kurs06.lsf - Stifte
Kurs07.lsf - Einfache Funktionen
Kurs08.lsf - Einfache Schleifen
Kurs09.lsf - Externe Dateien einbinden
Kurs10.lsf - Externe Dateien einbinden, Seitenausrichtung
Kurs11.lsf - Zellen
Kurs12.lsf - Tabellen
Kurs13.lsf - Tabellen
Kurs14.lsf - Bitmaps, Logo
Kurs15.lsf - Kopf / Fußzeilen
Kurs16.lsf - Druck in Datei DruckNT.int