

Bedienungsanleitung
SearARep
(Search And Replace)
(Stand: 16. Juli 2003)

Inhaltsverzeichnis

Einige Anmerkungen zu SearARep vorweg		9
Für welche Anwender ist SearARep geeignet?		9
Welche Daten kann SearARep bearbeiten?		10
Wie werden Dateien in SearARep verarbeitet?		10
Wie schnell arbeitet SearARep?		10
Kann SearARep etwas, was man nicht auch mit „regulären Ausdrücken“ und „Perlprogrammen“ erledigen kann?		11
<hr/>		
1	Installation des Programmes SearARep	13
1.1	Entpacken	13
1.2	Lizenzvereinbarung	13
1.3	Auswahl des Zielverzeichnisses	14
1.4	Sicherungsdateien erstellen	14
1.5	Verknüpfungen erstellen	14
<hr/>		
2	Erstaufruf des Programmes	15
2.1	Programmstart	15
2.2	Bedienungselemente und Anzeigen des Programmes	16
2.2.1	Kopfleiste des Programmfensters	16
2.2.1.1	Gültige Testversion mit Restlaufzeit	16
2.2.1.2	Testversion: Testzeit ist abgelaufen	16
2.2.1.3	Registrierte Programmversion	16
2.2.2	Testversion: Direkt nach Programmaufruf	17
2.2.3	Eingabemöglichkeiten des Programmes SearARep	18
2.2.3.1	Konvertiertabellen	18
2.2.3.2	Bedienung SearARep - Voreinstellungen - Verzeichnismaske setzen	19
2.2.3.3	Bedienung SearARep - Voreinstellungen - Konvertiertabelle auswählen	21
2.2.3.4	Bedienung SearARep - Datei(en) konvertieren	22
2.2.3.5	Bedienung SearARep - Verzeichnisbaum konvertieren	24
2.2.3.6	Bedienung SearARep - Programm beenden	25

Bedienungsanleitung: SearARep

3	Konvertiertabellen	26
3.1	Allgemeine Informationen zu den Konvertiertabellen	26
3.1.1	Format der Konvertiertabellen	26
3.1.1.1	Leerzeilen	27
3.1.1.2	Kommentare	28
3.1.1.2.1	einzeilige Kommentare	28
3.1.1.2.2	mehrzeilige Kommentare	28
3.1.1.3	Konvertiereregeln	29
3.1.1.3.1	Zeichen in Konvertiereregeln	32
3.1.1.3.2	Schreibweisen von Konvertiereregeln	33
3.1.1.3.3	Längenbegrenzungen von Such- und Ersatzbegriffen	34
3.2	Konvertiereregeln	35
3.2.1	Einfache Konvertierregel	35
3.2.2	Konvertierregel: Jokerfunktion	37
3.2.3	Konvertierregel: ganzzahlige Ziffer (Ziffernjoker)	38
3.2.4	Komplexe Funktionen	39
3.2.4.1	Syntax der <i>komplexen Funktion</i>	40
3.2.4.2	Beschreibung der einzelnen Funktionen	42
3.2.4.2.1	Ausgabefunktionen	42
	<i>print(u1,u2,u3,...,un)</i>	42
	<i>e(u1,u2,u3,...,un)</i>	43
3.2.4.2.2	Arithmetische Funktionen	44
	<i>add(n1 c1,n2 c2,n3 c3,...,nn cn)</i>	44
	<i>sub(n1 c1,n2 c2,n3 c3,...,nn cn)</i>	45
	<i>mul(n1 c1,n2 c2,n3 c3,...,nn cn)</i>	45
	<i>div(n1 c1,n2 c2,n3 c3,...,nn cn)</i>	46
	<i>av(n1 c1,n2 c2,n3 c3,...,nn cn)</i>	46
	<i>round(n1,n2,c3)</i>	47
	<i>str(n1,n2,n3)</i>	48
	<i>siv(c1 ,n2,n3,c4,c5)</i>	49
	<i>vat(n1)</i>	50
3.2.4.2.3	Datumsfunktionen	51
	<i>today(n1 , c2)</i>	51
	<i>todayts(n1)</i>	51
	<i>mdt(n1)</i>	52
	<i>md(c1)</i>	52
	<i>cald(c1,c2)</i>	53
	<i>woy(c1 d1)</i>	53
	<i>fts(c1,c2, n3)</i>	54

Bedienungsanleitung: SearARep

3.2.4.2.4	Zeitfunktionen	56
	<i>time()</i>	56
	<i>etime(c1,c2)</i>	56
	<i>ftime(c1,c2)</i>	57
	<i>sec()</i>	58
	<i>sec2time(n1)</i>	58
	<i>time2sec(c1)</i>	59
3.2.4.2.5	Textkettenmanipulationen (Stringmanipulationen)	60
	<i>atrim(c1, c2)</i>	60
	<i>ltrim(c1, c2)</i>	61
	<i>rtrim(c1, c2)</i>	61
	<i>fl(c1 n1,n2,c3)</i>	62
	<i>fr(c1 n1,n2,c3)</i>	62
	<i>chr(n1,n2,n3,...,nn)</i>	63
	<i>left(c1, n2)</i>	63
	<i>rightc1, n2)</i>	64
	<i>substr(c1, n2, n3)</i>	64
	<i>expand(c1 ,c2,n3)</i>	65
	<i>red(c1 ,c2,n3)</i>	66
	<i>convert(c1, c2)</i>	67
	<i>ins(c1, c2, n3)</i>	68
	<i>del(c1, n2, n3)</i>	68
	<i>slen(c1)</i>	69
	<i>sr(c1,c2,c3)</i>	69
3.2.4.2.6	Suchfunktionen	70
	<i>wisfl(c1,c2,n3)</i>	70
	<i>wisfr(c1,c2,n3)</i>	71
	<i>instr(c1,c2)</i>	71
3.2.4.2.7	Vergleichsfunktionen	72
	<i>bt(u1,u2,u3)</i>	72
	<i>eq(u1,u2)</i>	73
	<i>eeq(u1,u2)</i>	73
	<i>noteq(u1,u2)</i>	74
	<i>gt(u1,u2)</i>	74
	<i>gteq(u1,u2)</i>	75
	<i>ls(u1,u2)</i>	75
	<i>lseq(u1,u2)</i>	76
3.2.4.2.8	Bedingungsfunktionen	77
	<i>if(l1,u1,u2)</i>	77
	<i>ifn(l1,u1,u2)</i>	77

Bedienungsanleitung: SearARep

3.2.4.2.9	Dateifunktionen	78
	<i>rf(c1)</i>	78
	<i>sf(c1,c2)</i>	78
	<i>cofn(c1)</i>	79
	<i>lct(c1)</i>	79
3.2.4.2.10	Variablenfunktionen	80
	<i>Syntax der Variablen:</i>	80
	<i>sv(c1,u2)</i>	82
	<i>dv(c1)</i>	83
	<i>iev(c1)</i>	83
	<i>Inhalt einer Variablen auslesen</i>	84
	<i>Makroverarbeitung</i>	84
3.3	Verwendung von mehreren Konvertiertabellen	85
3.3.1	Aufgabenbezogene Konvertiertabellen	85
3.3.2	Auftragsbezogene Konvertiertabellen	85
3.3.3	Geschachtelte Konvertiertabellen	86
3.4	IPC-Kommandos (Inline Programming Command)	87
3.4.1	Einfache Konvertierregel	87
3.4.2	Wildcard- oder Jokerfunktionen	88
3.4.3	Ziffernjoker	89
3.5	Allgemeine Hinweise zu Konvertiertabellen und Konvertierregeln	90

Bedienungsanleitung: SearARep

4	Protokolldateien	91
5	Fehler	92
5.1	Fehlermöglichkeiten (Anwendungsfehler)	92
5.1.1	Fehler, die keine Fehlermeldung auslösen	92
5.1.2	Fehler, die zu Fehlermeldungen führen	93
5.1.3	Strategien zur Fehlersuche	95
5.1.4	Strategien zur Fehlervermeidung	97
6	Beispieldateien	98
6.1	Muster-01.knv	98
6.2	Muster-02.knv/Muster-02.txt	98
6.3	Muster-Arithmetik.knv/Muster-Arithmetik-2.knv/ Muster-Arithmetik.txt	99
7	SearARep in eine Vollversion umwandeln	100
8	Anpassung von Textpad für SearARep	102

Bedienungsanleitung: SearARep

Einige Anmerkungen zu SearARep vorweg

SearARep ist eine Anwendung, die mir in den letzten drei Jahren bei meinen vielfältigen Arbeiten im Bereich der **OCR-Verarbeitung** und der **automatischen Textaufbereitung** eine äußerst effektive Hilfe war.

Im Laufe dieser Zeit habe ich das Programm bei Bedarf immer wieder meinen erweiterten Anforderungen angepasst.

In diesem Rahmen ist das Programm intensiv getestet worden und hat eine sehr gute Praxiserprobung hinter sich.

Nachdem ich mich entschlossen habe, das Programm als **Shareware** allgemein zur Verfügung zu stellen, habe ich noch eine Reihe von Anpassungen an der Bedieneroberfläche vorgenommen und das Programm um einige Funktionen erweitert.

Auch diese Änderungen habe ich sorgfältig getestet. Sollte bei Ihnen dennoch ein Fehler (und kein Programm ist ohne Fehler!) auftreten, so informieren Sie mich bitte.

Für welche Anwender ist SearARep geeignet?

Die typischen Anwender für **SearARep** bearbeiten Daten und Texte, in denen bestimmte Arbeiten einen hohen Wiederholungsfaktor haben. Der Wiederholungsfaktor kann sich dabei auf **große Datenmengen** oder aber auch auf immer **wiederkehrende Aufgaben** beziehen.

SearARep ist so konzipiert, dass man, je Aufgabe **einmalig**, die zu erledigende Aufgabe **analysiert**, die **Verarbeitungsschritte** in einer Konvertiertabelle **festlegt** und diese **Arbeitsschritte** bei Bedarf **immer wieder** ablaufen lässt.

Die festgelegten Arbeitsschritte laufen **vollständig automatisch** ab und **brechen** in der Regel **auch im Fehlerfall nicht ab**. **Erkannte Fehler** werden **protokolliert** und am Ende eines Programmlaufes erfolgt ein **Fehlerhinweis**.

Der **Schwierigkeitsgrad** der Funktionen in den Konvertiertabellen geht von **sehr einfach** über **etwas schwieriger** (Jokerfunktionen) bis zu den **komplexen Funktionen**, die schon ein **gewisses Verständnis von Anwendungsprogrammierung** voraussetzen.

Experimentieren Sie mit dem Funktionsangebot. **90 Prozent** aller Aufgaben lassen sich schon mit **einfacheren Methoden lösen**.

Bedienungsanleitung: SearARep

Welche Daten kann SearARep bearbeiten?

SearARep ist für reine Textdateien im ANSI/ASCII-Format entwickelt worden. Grundsätzlich könnte **SearARep** auch viele andere Formate bearbeiten. Das Programm weist **kein Format** zurück. Sie als Anwender müssen allerdings schon genau wissen, für **welches Dateiformat** der Einsatz **sinnvoll** ist.

Ich verwende **SearARep** für die genannten reinen Textformate, für sogenannte **“tagged Formate”** (HTML, XML, spezielle Formate der DTP-Programme).

Wie werden Dateien in SearARep verarbeitet?

SearARep verarbeitet **einzelne** Dateien, **mehrere** Dateien in einem Verzeichnis oder auch **viele Dateien in einer Verzeichnisstruktur**.

SearARep verändert niemals Ihre Ausgangsdateien, sondern legt jede bearbeitete Datei in einer neuen Verzeichnisstruktur ab.

Sie können beispielsweise in **komplexen HTML-Dateistrukturen** in wenigen Sekunden in allen Dateien **absolute** Pfade durch **relative** Pfade ersetzen, bestimmte **Farbschemata ändern** und das Ganze in der erzeugten **Kopie der Dateistruktur** testen.

Wie schnell arbeitet SearARep?

Die **Verarbeitungsgeschwindigkeit** von SearARep hängt von der **Art der verwendeten Funktionen** und von deren **Anzahl** ab. Einfache Konvertiereregeln werden sehr schnell ausgeführt, bestimmte Jokerfunktionen erfordern schon etwas mehr Zeit, tief geschachtelte komplexe Funktionen und Makroverarbeitung können die Verarbeitungsgeschwindigkeit reduzieren.

Probieren Sie die **Arbeit mit SearARep** einfach aus. Ich denke, Sie werden von der Verarbeitungsgeschwindigkeit **angenehm überrascht** sein.

Bedienungsanleitung: SearARep

Kann SearARep etwas, was man nicht auch mit „regulären Ausdrücken“ und „Perlprogrammen“ erledigen kann?

Ich kann Ihnen diese Fragen **nicht** endgültig beantworten.

Der **routinierte** Umgang mit **„regulären Ausdrücken“** oder ein **perfektes Perlprogramm** führen sicherlich zu ähnlichen Ergebnissen wie die Arbeit mit **SearARep**.

Ich löse solche Aufgaben allerdings lieber mit **SearARep**.

Und wenn ich einmal unbedingt programmieren möchte, verwende ich die **komplexen Funktionen** oder erweitere **SearARep**.

Und, wenn Sie **nicht** programmieren wollen, verwenden Sie die **einfachen Konvertiereregeln** von **SearARep**, die Sie bei Bedarf mit den **Jokerfunktionen** erweitern.

Sollten Ihnen diese Möglichkeiten nicht reichen, stehen Ihnen über

50 komplexe Funktionen

mit den Möglichkeiten der **Makroprogrammierung** zur Verfügung.

Bei Wünschen, Anregungen oder gefundenen Fehlern schreiben Sie an

werner.perplies@weepee.de
Betreff: SearARep

Ich wünsche Ihnen ein **erfolgreiches** und **produktives** Arbeiten mit **SearARep**.

Garching, den 21. Juni 2003

Bedienungsanleitung: SearARep

1 Installation des Programmes SearARep

1.1 Entpacken

Die Auslieferung des Programmes *SearARep* erfolgt als selbstextrahierendes Archiv. Sie haben es entweder als Download oder auf einer CD erhalten.

Beginnen Sie die Installation des Programmes im Window-Explorer durch einen Doppelklick mit der Maus auf die Datei

SearARep.exe.

Danach werden die Dateien des Programmes entpackt und das Installationsprogramm gestartet:

Bestätigen Sie den Eröffnungsbildschirm mit

Weiter

1.2 Lizenzvereinbarung

Als nächstes sehen Sie die **Lizenzvereinbarung**, Sie können sie bei Bedarf zur Archivierung ausdrucken:

Drucken

Wenn Sie mit dieser Vereinbarung einverstanden sind, können Sie

Zustimmen

und die Installation wird fortgesetzt. Anderenfalls können Sie

Ablehnen

Die Installation wird dann an dieser Stelle abgebrochen.

Bedienungsanleitung: SearARep

1.3 Auswahl des Zielverzeichnisses

Standardmäßig wird das Programm in das Verzeichnis

c:\Programme\Perplies\SearARep

installiert. **Ändern** Sie ggf. den Verzeichnisnamen.

Weiter

1.4 Sicherungsdateien erstellen

Das Installationsprogramm schlägt Ihnen jetzt die Einrichtung eines Sicherungsverzeichnisses vor, hier werden eventuell schon vorhandene Dateien vor dem Überschreiben gerettet. Bestätigen Sie mit

Weiter

1.5 Verknüpfungen erstellen

Das Installationsprogramm schlägt Ihnen jetzt vor, verschiedene Verknüpfungen für ggf. alle Benutzer einzurichten.

Sie können hier einzelne Verknüpfungen abwählen und/oder die Installation auf den aktiven Benutzer beschränken.

Bestätigen Sie die Eingabe mit

Installieren

Danach wird das Programm installiert. Schließen Sie das Installationsprogramm mit

Beenden

Bedienungsanleitung: SearARep

2 Erstaufruf des Programmes

2.1 Programmstart

Starten Sie das Programm mit

Start → Programme → Perplies → SearARep

(gilt nur, wenn Sie diese Verknüpfung nicht abgewählt haben).

Das Programm richtet jetzt die notwendige Verzeichnisstruktur ein, erzeugt eine Voreinstellungsdatei und kennzeichnet das Programm als Testversion.

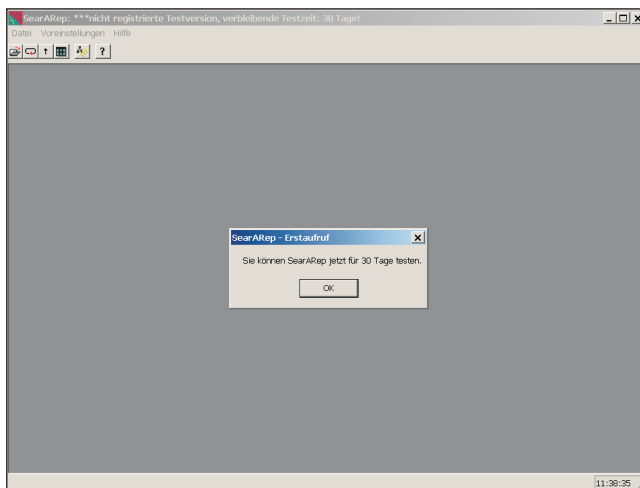


Abb. 1 SearARep nach dem Erstaufruf

Bestätigen Sie die Eingabe durch die *ENTER*-Taste oder klicken Sie mit der Maus auf *OK*.

Bedienungsanleitung: SearARep

2.2 Bedienungselemente und Anzeigen des Programmes

2.2.1 Kopfleiste des Programmfensters

Die Kopfleiste des Programmfensters informiert Sie über den Programmstatus:

2.2.1.1 Gültige Testversion mit Restlaufzeit

A screenshot of the SearARep header bar. It features a small red and green icon on the left. The text reads: "SearARep: ***nicht registrierte Testversion, verbleibende Testzeit: 30 Tage!".

Abb. 2 Kopfleiste: Testversion, Restlaufzeit 30 Tage

2.2.1.2 Testversion: Testzeit ist abgelaufen

A screenshot of the SearARep header bar. It features a small red and green icon on the left. The text reads: "SearARep: ***nicht registrierte Testversion, die Testzeit ist abgelaufen!***".

Abb. 3 Kopfleiste: Testversion, Testzeit ist abgelaufen

2.2.1.3 Registrierte Programmversion

A screenshot of the SearARep header bar. It features a small red and green icon on the left. The text reads: "SearARep: registriert auf die Firma Werner Perplies, EDV-Anwendungsberatung".

Abb. 4 Registrierte Programmversion

Bedienungsanleitung: SearARep

2.2.2 Testversion: Direkt nach Programmaufruf

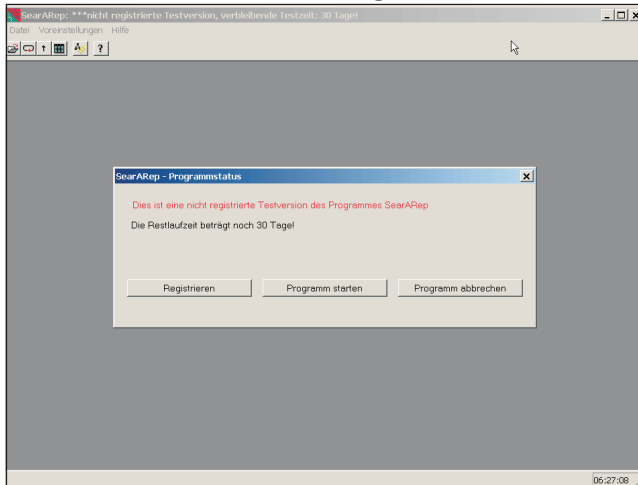


Abb. 5 Startdialog der Testversion

Bei jedem Aufruf der Testversion können Sie entscheiden, ob Sie

Das Programm registrieren wollen:

Umwandlung des Programmes in eine unbeschränkte Version durch Eingabe des erworbenen Schlüssels, siehe auch

Umwandlung des Programmes SearARep in eine Vollversion

(5 SearARep in eine Vollversion umwandeln, Seite 88)

oder

Das Programm starten wollen

oder

Den Programmlauf abbrechen wollen.

Bedienungsanleitung: SearARep

2.2.3 Eingabemöglichkeiten des Programmes SearARep

2.2.3.1 Konvertiertabellen

Absolute Voraussetzung für die die Arbeit mit **SearARep** ist die Einrichtung **mindestens einer Konvertiertabelle**.

Sie finden kommentierte Beispiele im Ordner

Programmordner\Beispiele\Konvertiertabelle -> Muster-....knv

Erstellen Sie die Konvertiertabellen außerhalb des Programmes **SearARep** mit Ihrem Lieblingseditor (Windowseditor, Wordpad oder jeder andere Editor, der in der Lage ist, reine ANSI/ASCII-Daten abzulegen). Ich persönlich **bevorzuge** das Programm **Textpad**, eine Testversion finden Sie unter

<http://www.textpad.com>

Im Verzeichnis

Programmordner\Textpad

finden Sie **drei Konfigurationsdateien**, mit deren Hilfe Sie das Programm **Textpad** für die

Erstellung von SearARep-Konvertiertabellen

optimieren **können**.

Weitere Hinweise zur **Anpassung von Textpad** finden Sie unter:

6 Anpassung von Textpad für SearARep, Seite 102

Informationen zur **Erstellung von Konvertiertabellen** finden Sie im Abschnitt:

3 Konvertiertabellen, Seite 26

Bedienungsanleitung: SearARep

**2.2.3.2 Bedienung SearARep
- Voreinstellungen - Verzeichnismaske setzen**

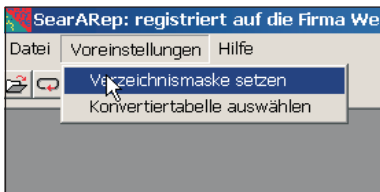


Abb. 6 Verzeichnismaske setzen (Menue)

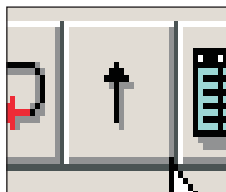


Abb. 7 (Knopf)

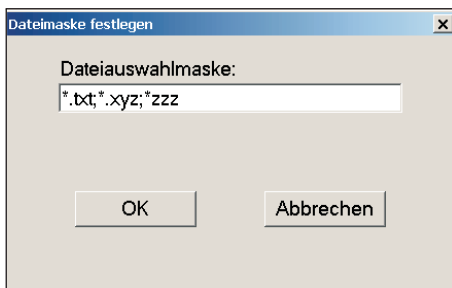


Abb. 8 Verzeichnismaske eingeben

Mit dieser Funktion können Sie festlegen, welche Dateien in der Dateiauswahl angezeigt werden und für welche Dateien die Funktion

Verzeichnisbaum konvertieren

gilt.

Die ausgewählte Einstellung wird in die Voreinstellungsdatei geschrieben.

Diese Maske gilt, bis sie wieder durch die Funktion

Verzeichnismaske setzen

geändert wird.

Bedienungsanleitung: SearARep

Innerhalb der Funktion

Dateien konvertieren

kann die Maske *temporär* für die einmalige Benutzung geändert werden.

Die folgende Maske

`*.txt;*.xyz;*.zzz;`

wählt zum Beispiel die folgenden Dateien aus:

Alle Dateien, die mit der Dateierweiterung **.txt** enden.

Alle Dateien, die mit der Dateierweiterung **.xyz** enden.

Alle Dateinamen, die einen beliebigen Text am Anfang enthalten,
mit **zzz** enden und keine Dateierweiterung haben.



Bedienungsanleitung: SearARep

2.2.3.3 Bedienung SearARep
- Voreinstellungen - Konvertiertabelle auswählen

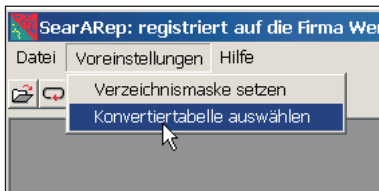


Abb. 9 Konvertiertabelle wählen (Menue)

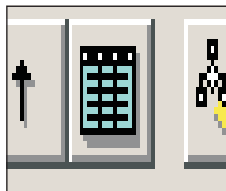


Abb. 10 (Knopf)

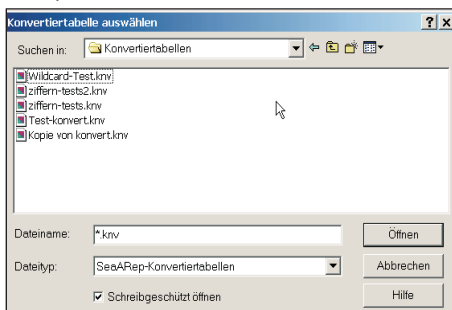


Abb. 11 Konvertiertabelle (Dateiauswahl)

Mit dieser Funktion können Sie festlegen, welche **Konvertiertabelle** für die Funktionen

Dateien konvertieren

und

Verzeichnisbaum konvertieren

benutzt werden soll.

Die jeweils **ausgewählte Tabelle** wird in der **Statusleiste** angezeigt.

Auch diese Einstellung bleibt so lange **erhalten**, bis sie **erneut geändert** wird.

Bedienungsanleitung: SearARep

Weitere Informationen zur Nutzung von Konvertiertabellen finden Sie unter

3 Konvertiertabellen, Seite 26

und **speziell** zur **Auswahl** von Konvertiertabellen:

3.3 Verwendung von mehreren Konvertiertabellen, Seite 85

2.2.3.4 Bedienung SearARep - Datei(en) konvertieren

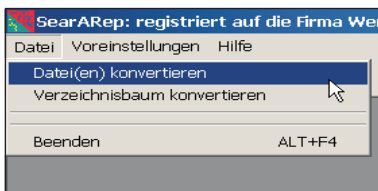


Abb. 12 Datei(en) konvertieren (Menue)

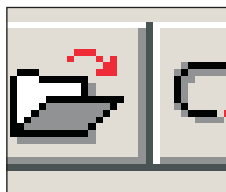


Abb. 13 (Knopf)

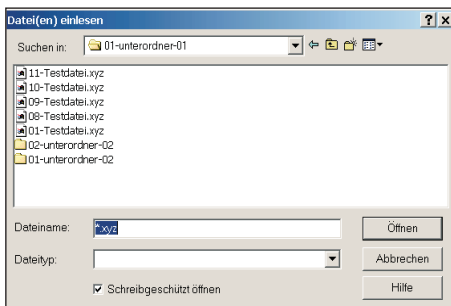


Abb. 14 Dateien konvertieren (Dateiauswahl)

Mit dieser Funktion können Sie eine oder mehrere Datei(en) in **einem** Ordner auswählen, die dann nach den Regeln der **eingestellten** Konvertiertabelle bearbeitet werden.

Nach *Bestätigung* der *Auswahl* wird die Konvertierung sofort durchgeführt. Über den Ablauf des Vorganges wird eine Protokolldatei erzeugt, in die

Bedienungsanleitung: SearARep

der *Zeitbedarf* der Konvertierung für jede Datei und eventuell auftretende Fehler eingetragen werden. Siehe dazu auch

4 Protokolldateien (Seite 87)

Mit der Auswahl der Dateien legen Sie zugleich automatisch das Zielverzeichnis für die konvertierten Dateien fest.

Im oben aufgeführten Beispiel wäre das ein Verzeichnis auf der **gleichen Ebene** wie das Ausgangsverzeichnis, mit dem **gleichem Namen**, vor den nur das Zeichen "_" eingefügt wurde.

Sollte dieses Ausgabeverzeichnis schon vorhanden sein, wird so lange ein weiteres _-Zeichen vorangestellt, bis sicher gewährleistet ist, dass das Zielverzeichnis noch nicht vorhanden ist.

Dieses Verfahren soll sicherstellen, dass **SearARep** niemals vorhandene Dateien überschreibt.

SearARep merkt sich Ihr ausgewähltes Ausgangsverzeichnis und wird Ihnen dieses Verzeichnis zukünftig bei weiteren Konvertierläufen als Ausgangsverzeichnis vorschlagen.

Bedienungsanleitung: SearARep

2.2.3.5 Bedienung SearARep - Verzeichnisbaum konvertieren

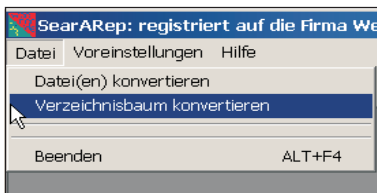


Abb. 15 Verzeichnisbaum konvertieren (Menue)

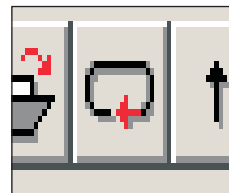


Abb. 16 (Knopf)

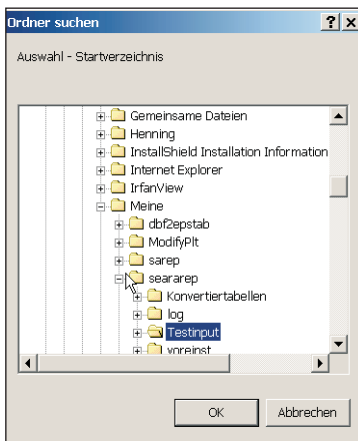


Abb. 17 Verzeichnisauswahl

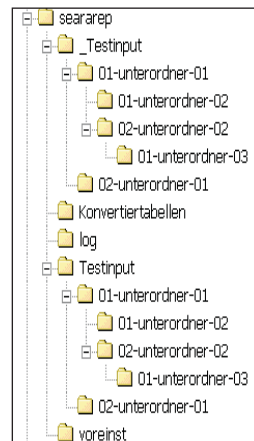


Abb. 18 Erzeugte Verzeichnisstruktur

Mit dieser Funktion können Sie einen ganzen Verzeichnisbaum konvertieren.

SearARep wählt mit Hilfe der voreingestellten Verzeichnismaske (z. B. *.txt) alle Dateien unterhalb des Ausgangsverzeichnisses aus, konvertiert sie nach den Regeln der voreingestellten Konvertiertabelle und legt sie unter **Beibehaltung** der Verzeichnisstruktur in einem **neu erzeugten Ausgangsordner** ab.

Der Ausgangsordner wird nach den gleichen Regeln erzeugt wie der Ausgangsordner für die Funktion *Datei(en) konvertieren*. So sollte auch hier sichergestellt sein, dass **SearARep** niemals Dateien versehentlich überschreibt.

Bedienungsanleitung: SearARep

2.2.3.6 Bedienung SearARep - Programm beenden

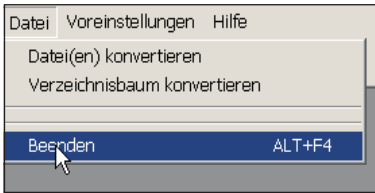


Abb. 19 Programm beenden
(Menue)

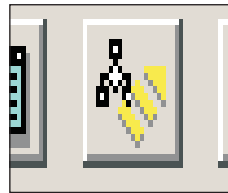


Abb. 20 (Knopf)

Mit dieser Funktion beenden Sie **SearARep**.

Bedienungsanleitung: SearARep

3 Konvertiertabellen

Die Konvertiertabellen für das Programm **SearARep** können mit einem beliebigen Texteditor erzeugt werden. Verwenden Sie am besten als Grundlage die mitgelieferten

Muster-0.knv

3.1 Allgemeine Informationen zu den Konvertiertabellen

3.1.1 Format der Konvertiertabellen

Konvertiertabellen sind ganz normale Textdateien im **ANSI-** oder **ASCII-**Format.

Konvertiertabellen haben die Dateierweiterung **.knv**.

Sie finden Mustertabellen unter

Programmordner\Konvertiertabellen\Muster-0.knv

Konvertiertabellen können aus den folgenden Elementen bestehen:

1. Leerzeilen
2. ein- oder mehrzeilige Kommentare
3. einzeilige Konvertierregeln
4. mehrzeilige Konvertierregeln (verbundene Zeilen)

Jedes **andere** Element wird, sofern es erkannt werden kann, als Fehler angesehen. Erkannte Fehler führen in der Regel **nicht zum Abbruch** des Programmes, **sondern** zu einem **Fehlereintrag** in der Protokolldatei und einem **Fehlerhinweis am Ende des Programmlaufes**.

Einige Fehler können systembedingt nicht erkannt werden und können in der erzeugten Datei zu unvorhergesehenen Ergebnissen führen.

Manche, **vermeintlich fehlerhaften** Ergebnisse, ergeben sich jedoch aus einer **falsch angewendeten** Konvertierregel.

Ich empfehle Ihnen daher **dringend**, die einzelnen verwendeten Regeln **sorgfältig zu testen** und das **Ergebnis zu überprüfen**.

Bedienungsanleitung: SearARep

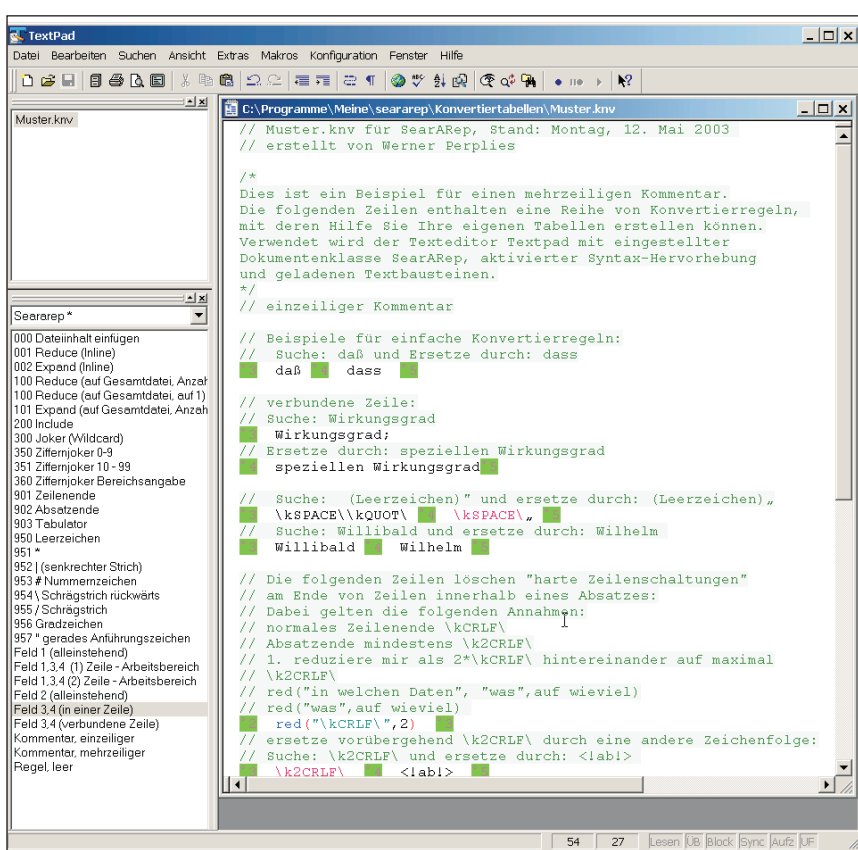


Abb. 21 Musterkonvertiertabelle in Textpad

3.1.1.1 Leerzeilen

Leerzeilen dürfen als einzige Zeichen Leerzeichen und Zeilenenden enthalten:

(Dezimalzeichen: 032, 010, 013; Hexadezimalzeichen: 20, 0D, 0A)

Mit Leerzeilen können Sie Ihre Konvertiertabelle gliedern.

Bedienungsanleitung: SearARep

3.1.1.2 Kommentare

Innerhalb einer Konvertiertabelle können Sie sowohl **einzeilige** als auch **mehrzeilige** Kommentare verwenden.

Kommentare dienen der **Dokumentation** und als **Erinnerungshilfe**. Umfangreiche Kommentare können die Einlesezeit einer Konvertiertabelle geringfügig verzögern. Sie verzögern jedoch nicht die eigentliche Verarbeitungsgeschwindigkeit.

Nutzen Sie deshalb ausgiebig die Möglichkeiten der Kommentierung.

3.1.1.2.1 einzeilige Kommentare

Einzeilige Kommentare beginnen immer mit der Zeichenfolge // und enden mit dem Zeilenende:

```
// Suche: (Leerzeichen)" und ersetze durch: (Leerzeichen)"
█ \kSPACE\kQUOTE\ █ \kSPACE\, █ // einfache Ersetzung
// Suche: Willibald und ersetze durch: Wilhelm
```

Abb. 22 Beispiele für einzeiligen Kommentar

Beim Einlesen einer Konvertiertabelle wird innerhalb einer Zeile alles ab der Zeichenfolge // überlesen. Verwenden Sie deshalb diese Zeichenfolge **niemals innerhalb** einer Konvertierregel, sondern verwenden Sie statt dessen die dafür vorgesehenen Spezialcodierungen.

Die Zeichenfolge // am Anfang der Zeile einer Konvertierregel **deaktiviert** die entsprechende Konvertierregel.

```
// °3 Willibald °4 Wilhelm °5
```

Abb. 23 Durch Kommentierung deaktivierte Konvertierregel

3.1.1.2.2 mehrzeilige Kommentare

Mehrzeilige Kommentare dienen in der Regel der ausführlicheren Dokumentation einer Konvertiertabelle. Sie beginnen mit der Zeichenfolge /* und enden mit der Zeichenfolge */. Verwenden Sie diese Zeichen bitte nur am Anfang einer Zeile.

```
/*
Dies ist ein Beispiel für einen mehrzeiligen Kommentar.
Die folgenden Zeilen enthalten eine Reihe von Konvertierregeln,
mit deren Hilfe Sie Ihre eigenen Tabellen erstellen können.
Verwendet wird der Texteditor Textpad mit eingestellter
Dokumentenklasse SearARep, aktivierter Syntax-Hervorhebung
und geladenen Textbausteinen.
*/
```

Abb. 24 Mehrzeiliger Kommentar

Bedienungsanleitung: SearARep

Sie können mit Hilfe dieser Zeichenfolgen auch mehrere Konvertiereregeln **gleichzeitig deaktivieren**.

Vergessene Kommentarmarkierungen führen allerdings oft zu völlig überraschenden Ergebnissen.

3.1.1.3 Konvertiereregeln

Konvertiereregeln bestehen aus einer Reihe von Felddaufrufen (zur Zeit maximal 4). Jedes Feld beginnt mit dem Zeichen ° (Grad), dem sofort eine Ziffer zwischen 1 und 5 folgen muss:

Die maximal mögliche **leere Konvertierregel** lautet:

```
// Vollständige leere Konvertierregel:
//   ° Feld 1   °   Feld 2   °   Feld 3   °   Feld 4
```

Abb. 25 Leere Konvertierregel

! jede Konvertierregel muss am Anfang einer Zeile beginnen **!**

Die Bedeutung der einzelnen Felder:

Feld 1:

kann Sonderkommandos enthalten:

```
// Beispiele Kommandos in Feld 1:
// Arbeitsbereich Zeile, Suche ... und ersetze durch
z ° Suchwort ° Ersatzwort °
// ° linclude ° 3Pfad+Dateiname ° 4 - Konvertiertabelle einfügen: Pfad und Dateiname
include ° c:\Programme\perplies\seararep\Konvertiertabelle\rohtext.knv °
// ipc, Befehl zur Kommandointerpretation ind Feld 3 und Feld 4
ipc ° ° hier Datei einfügen ° rf("c:\Texte\Beschreibung-017.txt") °
```

Abb. 26 Beispiele für Sonderkommandos in Feld 1

Diese **Sonderkommandos** sind zur Zeit **möglich**:

- include siehe Seite 85
- z siehe Seite 38
- ipc siehe Seite 87
- ipc+ siehe Seite 87
- ! siehe Seite 31

Bedienungsanleitung: SearARep

Feld 2:

kann **Funktionen** zur *komplexen Bearbeitung* der Datei und **Funktionen** enthalten, die auf die gesamten Daten eines Konvertiervorganges wirken sollen oder die keine unmittelbare Auswirkung auf den Textinhalt haben.

Funktionen, die der direkten Bearbeitung der Dateien dienen werden in der Form

Funktionsname(Parameter, Parameter,Parameter....)

aufgerufen. Die **Rückgabewerte** solcher Funktionen überschreiben den Inhalt der gerade bearbeiteten Datei.:

```
// komplexe Bearbeitung der Datei mit Kommandos und Funktionen
// Feld 2: (Beispiel)
// red("LEERZEICHEN",auf wieviel.1)
red(" ",1)
```

Abb. 27 komplexe Funktion red(), die Inhalt der Datei verändert, in Feld 2

Funktionen, die den **Inhalt** der bearbeiteten Datei **nicht verändern** sollen, müssen mit einem ! eingeleitet werden:

! Beachten Sie bei allen Anweisungen die Groß- und Kleinschreibung. **!**

Beispiele:

!sv("Variablennamen", "Inhalt der Variablen")

!dv("Variablennamen")

Dies sind in der Regel Funktionen, die **Speicherstellen** (Variablen) **verändern**, die erst **später** im Programm **benötigt** werden.

Siehe auch **Variablenfunktionen**, Seite 83

Feld 3:

Enthält **Suchbegriffe** oder notwendige Parameter (Werte) für Sonderkommandos:

Bedienungsanleitung: SearARep

Feld 4:

Enthält die Ersatzbegriffe

```
// Es werden nur die benötigten Felder verwendet:  
Suchwort Ersatzwort
```

Abb. 28 Such- und Ersatzbegriffe in Feld 3 und 4

In der **Standardeinstellung** wird **SearARep** die im Feld 4 enthaltenen Daten **an der Stelle** des **Suchbegriffes** in der zu verarbeitenden Datei als **Ersatz** ohne jede weitere Veränderung einfügen.

Abweichungen vom Standard:

Sie können dieses **Verhalten** durch verschiedene Faktoren **beeinflussen**:

!

Die **Verwendung** des Zeichens **"!" (Ausrufezeichen)** **einzeln** oder **am Ende eines Kommandos** in Feld 1 **verhindert die Ersetzung** des Suchbegriffes.

Eventuell **im Feld 4 vorhandene Inline-Kommandos** werden, wenn im **Feld 1 zusätzlich** die Kommandos **ipc** und **ipc+** vorhanden sind, für **jedes Vorkommen/letztes Vorkommen** des Suchbegriffes interpretiert und ausgeführt.

Die Rückgabe der höchsten Funktion wird verworfen.

Verwenden Sie dieses Kommando zum Beispiel zum Zählen von Suchbegriffen oder zur Berechnung von Werten.

Einsetzung von Wildcards/Jokern

Wenn Sie im Ersatzbegriff einmal oder mehrfach die Wildcardzeichen verwenden, werden diese vor der eigentlichen Ersetzung in der Datei durch die Daten ersetzt, die sich zwischen den Suchbegriffen befunden haben.

Siehe auch Jokerfunktion Seite 37, Ziffernjokerfunktion Seite 38

Bedienungsanleitung: SearARep

3.1.1.3.1 Zeichen in Konvertiereregeln

In den Konvertiereregeln können Sie alle Zeichen verwenden, die bei der Interpretation der Regeln keine Sonderfunktion haben.

Für manche Zeichen, die eigentlich problemlos verwendet werden könnten, gibt es ein Ersatzzeichen, das aus optischen Gründen verwendet werden kann.

Sonderzeichen:

Die folgenden Zeichen haben eine besondere Bedeutung und müssen deshalb in den *Such- und Ersatzbegriffen* speziell behandelt werden:

Zeichen	Ersatzzeichen	Name	
Zeilenende	\kCRLF\	Zeilenende Codierung: (dez: 13,10), (hex: 0D,0A)	1)
Absatzende	\k2CRLF\	Absatzende Codierung: (dez: 13,10,13,10), (hex:0D,0A,0D,0A)	1)
Tabulator	\kTAB\	Tabulator	2)
Leerzeichen	\kSPACE\	normales Leerzeichen	1)
°	\kDEGREE\	Gradzeichen	1)
*	\kASTERIX\	Sternzeichen	1)
	\kVLINE\	senkrechter Strich	1)
#	\kNCROSS\	Nummernzeichen	1)
\	\kBACKSLASH\	Schrägstrich rückwärts	1)
/	\kSLASH\	Schrägstrich vorwärts	1)
//	\k2SLASH\	Zwei Schrägstriche vorwärts	3)
“	\kQUOT\	einzelne Anführung, Zollzeichen	4)

Tabelle 1: Aufstellung der besonders zu behandelnden Zeichen

- 1) Diese Zeichen sind Bestandteil der Syntax der Konvertiereregeln
- 2) Dieses Zeichen der Konvertierregel wird vor Ausführung der Regel gelöscht.
- 4) Die Zeichenfolge “//” leitet **immer** einen Kommentar ein. Verwenden Sie deshalb **innerhalb einer Regel das Ersatzzeichen**.
- 3) Das Leerzeichen kann innerhalb einer Regel problemlos verwendet werden. Es sollte aber nicht als erstes oder letztes Zeichen innerhalb eines Feldes verwendet werden.

Bedienungsanleitung: SearARep

Nicht tastbare Zeichen oder **Zeichen mit Sonderfunktion im Editor** können Sie auch in der folgenden Form eingeben:

Dezimal: \d000\ bis \d255\
 Hexadezimal: \h00\ bis \hFF\

In den Konvertierregeln **direkt eingegebene Tabulatoren** dienen **der optischen Gestaltung** und werden **bei der Verarbeitung ignoriert**.

Wenn als **letztes Zeichen** einer Zeile oder **vor** einem einzeiligen Kommentar ein **Semikolon** eingegeben wird, wird beim Einlesen der Konvertiertabelle die Folgezeile mit der aktuellen Zeile **verbunden**.

Wenn Sie in einer Regel ein Semikolon verwenden wollen, verwenden Sie es **nicht** als **letztes Zeichen**. Geben Sie gegebenenfalls **zwei (;)** ein.

3.1.1.3.2 Schreibweisen von Konvertierregeln

Konvertierregeln können im Rahmen der vorgeschriebenen Syntax recht frei gestaltet werden:

Jede Regel kann **alle Felder** enthalten, nicht benutzte bleiben leer, z. B:

```
// Alle Felder werden eingegeben, nur zwei werden benutzt
Suchwort    Ersatzwort
```

Abb. 29 komplette Regel, zwei Felder benutzt

Sie können aber auch nicht benutzte Felder **weglassen**, gleichwertig wäre also:

```
// Es werden nur die benötigten Felder verwendet:
Suchwort    Ersatzwort
```

Abb. 30 nur benutzte Felder verwendet

Oder Sie verwenden **verbundene** Zeilen:

```
// verbundene Zeile:
// Suche: Wirkungsgrad
Wirkungsgrad:
// Ersetze durch: speziellen Wirkungsgrad
speziellen Wirkungsgrad
```

Abb. 31 Konvertierregel mit verbundenen Zeilen

Bedienungsanleitung: SearARep

Regeln, die über **mehrere Zeilen** gehen, müssen mit einem **Semikolon verbunden** werden. **Kommentarzeilen** benötigen das **Semikolon nicht**.

3.1.1.3.3 Längenbegrenzungen von Such- und Ersatzbegriffen

Die Länge der Such- und Ersatzbegriffe ist im Rahmen Ihres PC-Speichers prinzipiell **unbegrenzt**. Eine Überprüfung findet nicht statt.

Eine Überschreitung der Grenzen kann zu einem **Programmabbruch** führen.

Dies wird jedoch sicherlich extrem selten vorkommen.

Bedienungsanleitung: SearARep

3.2 Konvertiereregeln

3.2.1 Einfache Konvertierregel

Die einfachste Konvertierregel lautet:

Suche etwas und ersetze es, wenn es gefunden wird.

Einträge in der Konvertiertabelle:

```
// Suche: Luise isst ein Huhn
// Luise isst ein Huhn;
// Ersetze durch: Luise ist ein Huhn
// Luise ist ein Huhn
```

Abb. 32 Einfache Konvertierregel

Sie können auf diese Art auch **Zeichen löschen**:

```
Löschen
// Suche: Siehe auch Abschnitt 17.4.7. ... und ersetze durch: "nichts"
// Siehe auch Abschnitt 17.4.7.\kSPACE\
// Die folgende Zeile löscht doppelte Vorkommen des Wortes "der"
// Suche: " der der " und ersetze durch: " der "
// \kSPACE\der der\kSPACE\ \kSPACE\der\kSPACE\
```

Abb. 33 Beispiele: Konvertiereregeln zum Löschen

Beachten Sie bitte die **richtige Verwendung von Leerzeichen**. Wenn in Ihrem Text der folgende Satz vorkommen würde:

Der eine oder der andere Mitarbeiter fuhr mit dem Fahrrad durch den Wald zur Arbeit.

führt die folgende Konvertierregel zu einem Fehler:

```
*** ungenaue Konvertierregel ***
// Die folgende Zeile löscht doppelte Vorkommen des Wortes "der"
// Suche: " der der " und ersetze durch: " der "
// der der\kSPACE\ der\kSPACE\
// *** Ende der ungenauen Konvertierregel ***
```

Abb. 34 ungenaue Konvertierregel

Ergibt:

Der eine oder andere Mitarbeiter fuhr mit dem Fahrrad durch den Wald zur Arbeit.

Auch nicht schlecht, aber sicherlich **so** nicht gewollt!

Bedienungsanleitung: SearARep

Weitere Beispiele:

Am Anfang eines Absatzes etwas einsetzen - funktioniert nicht beim **ersten Absatz** und bei mehr als einer Zeilenschaltung hintereinander!

```
// Am Anfang eines Absatzes die Zeichenfolge "*****" einsetzen:
// | Suche: Absatzende und ersetze durch: "*****" + Absatzende
█ \k2CRLF█ █ \k2CRLF\*****█
```

Abb. 35 Einfügung am Absatzanfang (unvollständig)

Nach der Ausführung dieser Regel steht, außer vor dem ersten Absatz, am Anfang eines jeden Absatzes die Zeichenfolge *****. Das gilt auch für leere Absätze.

Unter der Annahme, dass leere Absätze in Ihrer Datei unerwünscht sind, können Sie die gestellte Aufgabe mit den folgenden Konvertiereregeln lösen:

```
// Noch einmal besser:
// Am Anfang eines Absatzes die Zeichenfolge "*****" einsetzen:
// Lösche alle Mehrfachvorkommen von Zeilenschaltungen:
// red("Zeilenschaltung", auf 2)
█ red("\k2CRLF", 2) █
// Suche: Absatzende und ersetze durch: "*****" + Absatzende
█ \k2CRLF█ █ \k2CRLF\*****█
// Füge am Anfang der Datei die Zeichenkette "*****" ein
// Suche: "gesamte Datei" und ersetze durch: "*****" + gesamte Datei
█ *****█ █
```

Abb. 36 Einfügung am Absatzanfang

Diese Aufgabe ist offensichtlich nicht mehr durch die *einfache Konvertierregel* lösbar.

Nutzen Sie deshalb zusätzlich die *komplexe Funktion red()* und die *Jokerfunktion* (Wildcardfunktion).

Eine Beschreibung dieser Konvertiereregeln finden Sie in den folgenden Abschnitten.

Die *einfache Konvertierregel* alleine bietet allerdings schon eine Vielzahl von Möglichkeiten, die sich erst nach einiger Zeit und einigen Experimenten voll erschließen.

So lassen sich viele Probleme auch dadurch lösen, dass Sie zuerst eine temporäre Konvertierung machen, die Sie nach Durchführung der eigentlichen Konvertieraufgabe wieder rückgängig machen:

```
// Ersetze alle einfachen ">" durch "!!", aber keinesfalls ">>"
// Suche ">>" und ersetze durch: "?merk ich mir?"
█ >> █ ?merk ich mir? █
// Suche ">" und ersetze durch "!!"
█ > █ !! █
// Suche "?merk ich mir?" und ersetze durch ">>"
█ ?merk ich mir? █ >> █
```

Abb. 37 Konvertierung mit temporärer Ersetzung

Bedienungsanleitung: SearARep

3.2.2 Konvertierregel: Jokerfunktion

Mit der Konvertierregel **Jokerfunktion** können Sie beliebige Begriffe (Textmuster) suchen.

Zur Zeit ist genau ein Joker in einer Konvertierregel erlaubt:

Der Joker wird als Zeichenfolge `|*\|` oder `|*Variablenname|` aufgerufen.

Der Joker steht für den Befehl

Suche irgendetwas und merke es.

Beispiele:

```
// Suche: "Suche |*\| und Merke es" und ersetze durch: "Suche das Buch und merke"
Suche |*\| und Merke es   Suche das Buch und merke
```

Abb. 38 Konvertierregel Jokerfunktion

Diese Regel auf den Satz

Suche irgendetwas und merke es

angewandt, ergibt

Suche das Buch und merke irgendetwas

weitere Beispiele:

```
// Dieses Beispiel fügt "<absatz>" am Anfang eines Absatzes und
// "</absatz>" am Ende eines Absatzes ein:
// Suche: Joker+Absatzende und ersetze durch: <absatz>|*\|</absatz> + Zeilenende
|*\| \k2CRLF\   <absatz>|*\|</absatz>\kCRLF\

// Dieses Beispiel fügt "<dokument> + Zeilenende" am Anfang einer Datei und
// "</dokument> + Zeilenende" am Ende einer Datei ein.
// Suche: "gesamte Datei" und ersetze durch "<dokument> + Zeilenende +
// gesamte Datei + </dokument> + Zeilenende"
|*\|   <dokument>\kCRLF\|*\|</dokument>\kCRLF\

// Diese Regel löscht alles von Suchbegriff 1 bis Suchbegriff 2
// Suche "von Suchbegriff 1 bis Suchbegriff 2" und ersetze durch: "nichts"
Suchbegriff 1|*\|Suchbegriff 2

// Diese Regel stellt Befehle um
<absatz><fett>|*\|</fett></absatz>\kCRLF\   @fett = |*\| \k2CRLF\

// Vertauschung mit Jokerfunktion:
// Suche "EUR beliebig + Zeilenende" und ersetze durch "beliebig EUR Zeilenende"
EUR |*\| \kCRLF\   |*\| EUR \kCRLF\
```

Abb. 39 verschiedene Anwendungen der Jokerfunktion

Auch hier gilt, experimentieren Sie mit den Möglichkeiten der **Jokerfunktion**.

Bedienungsanleitung: SearARep

Normalerweise ist der **Wirkungsbereich** jeder Konvertierregel die gesamte Datei.

Dies kann immer dann zu Problemen führen, wenn Sie davon ausgehen, dass Suchbegriff_1 **und** Suchbegriff_2 **paarweise** vorhanden sind.

Wenn dann diese **Voraussetzung nicht erfüllt** ist, kann es zu **unerwarteten Ergebnissen** kommen.

Deshalb kann es sinnvoll sein, den **Suchbereich** immer dann auf eine Zeile zu begrenzen, wenn Sie voraussetzen können, dass Suchbegriff_1 und Suchbegriff_2 immer innerhalb einer Zeile (begrenzt durch \kCRLF) vorkommen müssen. Dies geschieht durch das Kommando **z** in **Feld 1**:

```

// Begrenzung des Suchbereiches auf eine Zeile
// Suche: " beliebig " und ersetze durch:
z \kSPACE\ \kQUOT\ \kQUOT\ \kSPACE\ " "

```

Abb. 40 Suchbereichsbegrenzung auf eine Zeile

3.2.3 Konvertierregel: ganzzahlige Ziffer (Ziffernjoker)

Mit der Konvertierregel **ganzzahlige Ziffer** können Sie beliebige ganzzahlige Ziffern (Ziffernjoker) suchen.

Zur Zeit ist genau ein Ziffernjoker in einer Konvertierregel erlaubt:

Der Ziffernjoker kann auf verschiedene Arten im Suchbegriff aufgerufen werden:

Stellenbezogene Eingabe:

- |#| *suche beliebige Ziffer zwischen 0 und 9*
- ##| *suche beliebige Ziffer zwischen 10 und 99*

Bereichsbezogene Eingabe:

- |#|17-345| *suche beliebige Ziffer zwischen 17 und 345*
- |#|-123| *suche beliebige Ziffer zwischen 100 und 123*
- |#|1000-| *suche beliebige Ziffer zwischen 1000 und 9999*

Die gemerkte Zahl kann durch das # im Ersatzbegriff (**auch mehrfach**) eingefügt werden.

Bedienungsanleitung: SearARep

Beispiele:

```
// Beispiele für Ziffernjoker:
// Suche: §§ 17,18,19,20,21,22,23,24,25,26,27 XyzGB und ersetze durch
// $ 17 XyzGB (gestrichen), $ 18 XyzGB (gestrichen), $ 19 XyzGB (gestrichen)...
// $ #17-27$ # (gestrichen)

// Suche die Ziffern 1 - 34 am Zeilenanfang und füge ein neues Format ein:
// Suche ... und ersetze durch
// \kRLF\#1-34\ \kRLF\@zahl = #

// Suche: .Ziffer von 0-9. und ersetze durch .00(0-9).
// Suche Ziffern 0 - 9 und ersetze durch
// .#.#.

// Suche Ziffern 1000 - 9999 und ersetze durch ****:
// ####\ \kRLF\ ****
```

Abb. 41 Beispiele für die Anwendung von Ziffernjoker

3.2.4 Komplexe Funktionen

Über die bereits beschriebenen Möglichkeiten hinaus gibt es noch die Möglichkeit, innerhalb von Ersatzbegriffen Aufrufe für kleine Programme einzugeben, die ich im Folgenden *komplexe Funktionen* nenne.

Aus Geschwindigkeitsgründen muss der Konvertierregel mit speziellen Befehlen in Feld 1 mitgeteilt werden, dass *komplexe Funktionen* auszuwerten sind:

Dies geschieht durch die Kommandos **ipc**, **zipc** und deren jeweilige **Plusversion**. Beide Kommandos bewirken die Auswertung von *komplexen Funktionen* und unterscheiden sich nur im Suchbereich der Konvertierregel. **zipc** schränkt den Suchbereich auf eine Zeile ein (siehe auch Seite 38).

Siehe hierzu auch: **3.4 IPC-Kommandos**, Seite 87

Eine *komplexe Regel* besteht aus einem **Namensteil** (Kommando), der festlegt, **was** zu tun ist, und einem **Parameterteil** (Werteteil), der festlegt, **womit** etwas zu tun ist. Die Anzahl und die Art der Parameter hängen vom verwendeten Kommando ab.

Jeder Parameter kann wieder eine *komplexe Funktion* sein.

Eine *komplexe Funktion* kann einen Wert zurückgeben. Der **Datentyp** ist **abhängig** von der **verwendeten Funktion**.

Erfordert eine komplexe Funktion als Wert etwas anderes als eine Textkette, so muss das Ergebnis der im Parameter verwendeten Funktion in den notwendigen Datentyp **umgewandelt** werden.

Ist die Umwandlung nicht möglich, wird ein Fehler erkannt, der Fehler in die Protokolldatei geschrieben und ein Standardwert zurückgegeben.

Bedienungsanleitung: SearARep

3.2.4.1 Syntax der komplexen Funktion

Eine *komplexe Funktion* wird in der Form

Funktionsname(|Parameter1, Parameter2,Parameter3,.....|)

aufgerufen.

Parameter, die in senkrechten Strichen eingeklammert sind, können weggelassen werden. Diese Werte werden dann durch einen Standardwert ersetzt. Näheres erfahren Sie bei den einzelnen Kommandos.

Für die Parameter können Sie verschiedene Datentypen verwenden:

Datentyp	Beispiel (Schreibweise)	Prefix in der Funktionsbeschreibung
Textkette (String)	„Dies ist ein String“	c
Numerisch (Zahl)	3, 3425.17, 128	n
Logisch	true, false, ja, nein, t, f, j, n	l
Datum	“01.01.0001”	d
Unbestimmt	“txt”, 34, “true”, “17.04.1923”	u

Tabelle 2: Datentypen in komplexen Funktionen

In den Funktionsbeschreibungen finden Sie deshalb die folgende Schreibweise:

Funktionsname(c1,n2,l3,...)

Es kann auch Funktionen ohne einen Parameter geben

Funktionsname()

Verwenden Sie bitte für die **äußerste** komplexe Funktion in einem Ersatzbegriff die folgende Schreibweise:

|\$C Funktionsname(c1,c2,n3) \|

Bei einer geschachtelten komplexen Funktion könnte das so aussehen:

|\$C Funktionsname(c1,Funktionsname(n1),n3) \|

Bedienungsanleitung: SearARep

Beispiel für eine komplexe Funktion:

```
// Beispiel für eine komplexe Funktion:  
// Arbeitsbereich Gesamtdatei:  
// Dieses Beispiel  
ipc !?Linie einfügen?!\\kCRLF\ ; // sucht in der gesamten Datei  
// "!?Linie einfügen?!\\kCRLF\  
// und ersetzt es durch den Inhalt der Datei linie.txt,  
// in der zuvor alle -Zeichen durch 80 -Zeichen ersetzt wurden.  
[SC_expand(rf("C:\Bausteine\linie.txt"), " ", 80)]
```

Abb. 42 Beispiel für eine komplexe Funktion

Bedienungsanleitung: SearARep

3.2.4.2 Beschreibung der einzelnen Funktionen**3.2.4.2.1 Ausgabefunktionen****print(u1,u2,u3,...,un)**

Die Funktion **print** erzeugt aus den einzelnen Parametern eine Textkette und fügt sie an der aufgerufenen Stelle ein. Bei Bedarf werden die einzelnen Parameter in den Datentyp **STRING** umgewandelt.

Parameter:**u1 ... un** → beliebige Anzahl beliebiger Datentypen

```
print("Eva ", "isst ", "ein Huhn") ➔ "Eva isst ein Huhn"  
print(1,2,3,4,5,6,7,8,9,0) ➔ "1234567890"  
print(mul(17,4), " * ", 23, " = ", mul(mul(17,4), 23))  
➔ "68 * 23 = 1564"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

e(u1,u2,u3,...,un)

Die Funktion **e(Empty, Leerfunktion)** verarbeitet gültige Parameter und gibt keinen Wert zurück.

Parameter:

u1 ... un → beliebige Anzahl beliebiger Datentypen

```
e(sv("$I_Anzahl", add($I_Anzahl, 1))) ➔ keine Rückgabe
```

Diese Funktion gibt keinen Wert zurück.

Einsatzgebiet:

ICP-Kommandos, in denen für jeden Einzelschritt Funktionen ausgeführt werden sollen, deren Ausgabe nicht direkt in den Daten erscheinen soll:

Zum Beispiel: Häufigkeit zählen, Daten sammeln, Summen bilden

Bedienungsanleitung: SearARep

3.2.4.2.2 Arithmetische Funktionen**add(n1|c1,n2|c2,n3|c3,...,nn|cn)**

Die Funktion **add** (Addition) addiert die Zahlen von **n1|c1** bis **nn|cn** und gibt das Ergebnis als Zahl zurück.

Der **Datentyp String** wird zuvor in einen **Zahlendatentyp** umgewandelt.

Parameter:

n1 ... nn → beliebige Anzahl numerischer Datentypen

c1 ... cn → beliebige Anzahl Strings, die Zahlen enthalten
beliebige Mischung beider Datentypen

Leerzeichen werden im Beispiel durch **Punkte** abgebildet.

<code>add(17,23,23.17,33,44.50)</code>	➔	140.67
<code>add(17,-23,23.17,33,-44.50)</code>	➔	5.44
<code>str(add(17,23,23.17,33),-1,0)</code>	➔	"96"
<code>str(add(17,23,23.17,33),5,0)</code>	➔	"...96"

Die Funktion gibt eine Zahl zurück.

Bedienungsanleitung: SearARep

sub(n1|c1,n2|c2,n3|c3,...,nn|cn)

Die Funktion **sub** (Subtraktion) subtrahiert die Zahlen von **n1** bis **nn** und gibt das Ergebnis als Zahl zurück.

Der **Datentyp String** wird zuvor in einen **Zahlendatentyp** umgewandelt.

Parameter:

- n1 ... nn** → beliebige Anzahl numerischer Datentypen
- c1 ... cn** → beliebige Anzahl Strings, die Zahlen enthalten
beliebige Mischung beider Datentypen

Leerzeichen werden im Beispiel durch **Punkte** abgebildet.

```
sub(17,23,23.17,33,44.50)      ➔ -106.67
sub(17,-23,23.17,33,-44.50)   ➔ 28.33
str(sub(17,23,23.17,33),-1,0) ➔ "-62"
str(add(17,23,23.17,33),5,0)  ➔ "..-62"
```

Die Funktion gibt eine Zahl zurück.

mul(n1|c1,n2|c2,n3|c3,...,nn|cn)

Die Funktion **mul** (Multiplikation) multipliziert die Zahlen von **n1** bis **nn** und gibt das Ergebnis als Zahl zurück.

Der **Datentyp String** wird zuvor in einen **Zahlendatentyp** umgewandelt.

Parameter:

- n1 ... nn** → beliebige Anzahl numerischer Datentypen
- c1 ... cn** → beliebige Anzahl Strings, die Zahlen enthalten
beliebige Mischung beider Datentypen

Leerzeichen werden im Beispiel durch **Punkte** abgebildet.

```
mul(17,23,23.17)      ➔ 9059.47
mul(17,-23,23.17)     ➔ -9059.47
str(mul(17.5,23,33),-1,0) ➔ "13282"
str(mul(17,23.17),5,0) ➔ "..393"
```

Die Funktion gibt eine Zahl zurück.

Bedienungsanleitung: SearARep

div(n1|c1,n2|c2,n3|c3,...,nn|cn)

Die Funktion **div** (Division) dividiert die Zahlen von **n1** bis **nn** und gibt das Ergebnis als Zahl zurück.

Der **Datentyp String** wird zuvor in einen **Zahlendatentyp** umgewandelt.

Parameter:

n1 ... nn → beliebige Anzahl numerischer Datentypen

c1 ... cn → beliebige Anzahl Strings, die Zahlen enthalten
beliebige Mischung beider Datentypen

Leerzeichen werden im Beispiel durch **Punkte** **abgebildet**.

```
div(17,23,23,33)           ➔ 0.00
str(div(17,23,23,33),11,9) ➔ "0,000973821"
str(mul(div(17,100),16),-1,2) ➔ "2,72"
```

Die Funktion gibt eine Zahl zurück.

av(n1|c1,n2|c2,n3|c3,...,nn|cn)

Die Funktion **av** (Average, Durchschnitt) berechnet den Durchschnitt der Zahlen von **n1** bis **nn** und gibt das Ergebnis als Zahl zurück.

Der **Datentyp String** wird zuvor in einen **Zahlendatentyp** umgewandelt.

Parameter:

n1 ... nn → beliebige Anzahl numerischer Datentypen

c1 ... cn → beliebige Anzahl Strings, die Zahlen enthalten
beliebige Mischung beider Datentypen

Leerzeichen werden im Beispiel durch **Punkte** **abgebildet**.

```
av(17,23,23,33)           ➔ 24
```

Die Funktion gibt eine Zahl zurück.

Bedienungsanleitung: SearARep

round(n1,n2,c3)

Die Funktion **round** (Runden) rundet die Zahl **n1** auf **n2** Stellen und gibt je nach Inhalt von **c3** eine Zahl oder einen String zurück.

Parameter:

n1 → die zu rundende Zahl

n2 → die Anzahl der Stellen

c3 → entweder "c" (Rückgabe String) oder "n" (Rückgabe Zahl)

Dabei gilt:

letzte Stelle < 5, abrunden, letzte Stelle => 5, aufrunden

<code>round(17.234518, 5, "n")</code>	➔ 17.23452 (Zahl)
<code>round(17.234518, 4, "c")</code>	➔ "17,2345" (String)
<code>round(17.234518, 3, "n")</code>	➔ 17.235 (Zahl)

Diese Funktion gibt entweder einen String oder eine Zahl zurück.

Beachten Sie hierzu auch die Funktion **str()** auf der Seite 48

Bedienungsanleitung: SearARep

str(n1,n2,n3)

Die Funktion **str** (String) wandelt eine Zahl in einen String um. Dabei gilt:

Die umzuwandelnde Zahl **n1**, wird in einen String mit **n2** Zeichen **Gesamtlänge** und **n3** Dezimalstellen umgewandelt. Der String wird ggf. von links mit **Leerzeichen aufgefüllt**.

Parameter:

n1 → die umzuwandelnde Zahl

n2 → die Gesamtlänge des Ausgabestrings

n3 → die Anzahl der Dezimalstellen (**nicht gerundet!**)

Sonderfälle:

n2 = -1, die Zahl wird nicht mit Leerzeichen aufgefüllt.

n2 zu klein gewählt, die Zahl wird durch das/die Zeichen * ersetzt!

n3 = -1, es wird die interne Anzahl der Dezimalstellen verwendet.

Für die Darstellung des Beispiels:

Leerzeichen werden zur **besseren Darstellung** durch einen **Punkt ersetzt**:

<code>str(234.234567, 12, -1)</code>	➔	<code>"..234,234567"</code>
<code>str(234.234567, 12, 2)</code>	➔	<code>".....234,23"</code>
<code>str(234.234567, -1, -1)</code>	➔	<code>"234.234567"</code>

Die Funktion gibt einen String zurück.

Beachten Sie hierzu auch die **round**-Funktion auf der Seite 47

Bedienungsanleitung: SearARep

siv(c1|,n2,n3,c4,c5)

Die Funktion **siv** (StoreIncrementVariable) ist eine Zählfunktion. Je nach aufgerufenen Parametern wird entweder die Variable c1 vorbelegt oder um die Zahl 1 erhöht, abgespeichert und zurück gegeben.

Parameter:

- c1** → muss einen gültigen Variablennamen enthalten. **Erlaubt** sind nur Variablen des Datentyps **I (Integer, ganze Zahlen)**, falls die Variable noch **nicht existiert**, wird sie **eingrichtet** und mit **Null vorbelegt**.
- n2** → legt den **Startwert** des Zählers fest. Wenn **n2** bei Erstaufuf einer Variablen weggelassen wird, wird Null angenommen.
- n3** → definiert die Gesamtlänge des zurückgegebenen Zählers. Wenn **n3** bei Erstaufuf einer Variablen weggelassen wird, wird Null angenommen. Bei **Eingabe** der Länge **Null** wird der Zähler **ohne Füllzeichen von links** zurückgegeben.
- c4** → **bestimmt das/die Füllzeichen**. Wenn **c4** bei Erstaufuf weggelassen wird, wird die **Null** als Füllzeichen (Vornull) **verwendet**.
- c5** → kann ein beliebiger String sein, der vor dem Zähler eingefügt wird. Wird **c5** bei Erstaufuf weggelassen, wird nichts vor dem Zähler weggelassen.

! Parameter dürfen nur von hinten weggelassen werden. **!**

```
siv("$I_RN",999,4,"0","2003/") ➔ "2003/0999"
// nochmaliger Aufruf ohne die Parameter 2- 5
siv("$I_RN") ➔ "2003/1000"
// nochmaliger Aufruf mit neuem Startwert:
siv("$I_RN",2001) ➔ "2003/2001"
// nochmaliger Aufruf ohne die Parameter 2- 5
siv("$I_RN") ➔ "2003/2002"
```

Die Funktion gibt einen String zurück.

Bedienungsanleitung: SearARep

3.2.4.2.3 Datumsfunktionen

today(|n1|,|c2|)

Die Funktion today(heute) ermittelt ein Datum:

Parameter:

- n1** → ganze positive oder negative Zahl (Tage), weggelassen: 0
- c2** → "c" Rückgabe String, "n" → Rückgabe Datum, weggelassen: Datum

Annahme: aktuelles Datum ist: 5. Juni 2003

today()	➔ 05.06.2003 (Datum)
today(10)	➔ 15.06.2003 (Datum)
today(-3)	➔ 02.06.2003 (Datum)
today(17, "c")	➔ "22.06.2003" (String)

Der benötigte Datentyp (Datum,String) hängt von der Weiterverarbeitung ab.

todayts(|n1|)

Die Funktion **todayts**(Today-Timestamp, heute-Uhrzeit) ermittelt ein spezielles Datumsformat, bestehend aus Datum und Uhrzeit. Dieses Format wird zum Beispiel in SQL-Datenbanken verwendet.

Parameter:

- n1** → ganze positive oder negative Zahl (Tage), weggelassen: 0

Annahme: aktuelles Datum ist: 5. Juni 2003, 17:23:56

todayts()	➔ "2003-06-05 17:23:56"
todayts(10)	➔ "2003-06-15 17:23:56"
todayts(-3)	➔ "2003-06-02 17:23:56"

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

mdt(|n1)

Die Funktion **mdt**(MakeDateToday) ermittelt das aktuelle Datum, addiert oder subtrahiert ggf. Tage und gibt das Datum in der Form **jjjjmmtt** als String zurück.

Parameter:

n1 → ganze positive oder negative Zahl (Tage), weggelassen: 0

Annahme: aktuelles Datum ist: 5. Juni 2003

<code>mdt()</code>	➔ "20030605"
<code>todayts(10)</code>	➔ "20030615"
<code>todayts(-3)</code>	➔ "20030602"

Die Rückgabe ist ein String.

md(c1)

Die Funktion **md**(MakeDate) konvertiert ein Datum der Form **dd.mm.jjjj** in die Form **jjjj.mm.tt**.

Parameter:

c1 → ein gültiges Datum in der Form **tt.mm.jjjj**

<code>md("06.06.2003")</code>	➔ "20030606"
-------------------------------	--------------

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

cald(c1,c2)

Die Funktion **cald**(CalculateDays, Tage berechnen) berechnet die Differenz zwischen zwei gültigen Daten in Tagen..

Parameter:

c1 → ein gültiges Datum in der Form **tt.mm.jzzj**

c2 → ein gültiges Datum in der Form **tt.mm.jzzj**

`md("06.06.2003","16.06.2003")` → 10

Die Rückgabe ist eine Zahl.

woy(|c1|d1|)

Die Funktion **woy**(WeekOfYear, Kalenderwoche) berechnet die **Kalenderwoche** eines Datums

Parameter:

c1 → ein gültiges Datum in der Form "**tt.mm.jzzj**"

d1 → die Rückgabe einer Datumsfunktion
ohne d1 oder **c1** das aktuelle Tagesdatum

Annahme: aktuelles Tagesdatum ist 6. Juni 2003

`woy()` → "23"
`woy(today())` → "23"
`woy("06.06.2003")` → "23"

Die Rückgabe ist einString

Bedienungsanleitung: SearARep

fts(c1,c2,|n3|)

Die Funktion **fts**(FormatTimeStamp, formatiert Datum/Zeitmarken) ermöglicht spezielle Formatierungen von Datumsangaben und Zeitmarken.

Parameter:

c1 → ein gültiges Datum in der Form **tt.mm.jjjj**

oder

c1 → eine gültige Zeitmarke in der Form **jjjj.mm.tt 00:00:00**

c2 → ein gültiger Formatstring, siehe unten

n3 → eine ganze positive/negative Zahl (Tage), weggelassen 0

Der **Formatstring** kann aus verschiedenen Teilen bestehen:

Tage:

d/t → Tagesdatum ein- oder zweistellig
dd/tt → Tagesdatum zweistellig
ddd/ttt → Wochentag auf drei Zeichen gekürzt
dddd/tttt → Wochentag ausgeschrieben

Monate:

m → Monat als Zahl ein- oder zweistellig
mm → Monat als Zahl zweistellig
mmm → Monat auf drei Zeichen gekürzt
mmmm → Monat ausgeschrieben

Jahre:

j/y → Jahr einstellig
jj/yy → Jahr zweistellig, ggf. mit Vornull
jjjj/yyyy → Jahr vierstellig

Sonderzeichen:

: → nur innerhalb eines Datums erlaubt
: → nur innerhalb einer Zeitangabe erlaubt
'beliebige Zeichen' → innerhalb der Zeichen ' ' können Sie beliebige **testbare** Zeichen (keine Ersatzzeichen, keine komplexen Funktionen), eingeben.

Bedienungsanleitung: SearARep

Zeitangabe:

Eine Zeitangabe innerhalb eines Formatstrings wird in die Zeichenfolge |Zeitangabe| eingeklammert.

Stunden:

- h → Stunde einstellig
- hh → Stunde zweistellig

Minuten:

- m → Minuten einstellig
- mm → Minuten zweistellig

Sekunden:

- s → Sekunden einstellig
- ss → Sekunden zweistellig

Sonderzeichen:

- :
 - 'beliebige Zeichen'
- Trennzeichen
 - innerhalb der Zeichen '' können Sie beliebige **tafbare** Zeichen (keine Ersatzzeichen, keine komplexen Funktionen), eingeben,

```
fts("06.06.2003","tttt', den 't.' 'mmmm' 'jjjj")
➔ "Freitag, den 6. Juni 2003"
fts("06.06.2003","tttt', den 't.' 'mmmm' 'jjjj",2)
➔ "Sonntag, den 8. Juni 2003"
fts("2003-06-06 17:23:00","'am 'd.' 'mmmm' um '|h' Uhr'|")
➔ "am 6. Juni um 17 Uhr"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

3.2.4.2.4 Zeitfunktionen**time()**

Die Funktion **time()** gibt die aktuelle Zeit im Format **hh:mm:ss** zurück.

Parameter:

kein Parameter

Annahme: Die aktuelle Uhrzeit ist **15:34:17**

```
time() ➔ "15:34:17"
```

Die Rückgabe ist ein String.

etime(c1,c2)

Die Funktion **etime**(ElapTime, Zeitdifferenz) ermittelt die **Differenz** zwischen **zwei gültigen Zeitstrings**.

Parameter:

c1 → gültiger Zeitstring der Form "**hh:mm:ss**"

c2 → gültiger Zeitstring der Form "**hh:mm:ss**"

Annahme: Die aktuelle Uhrzeit ist **16:01:07**

```
etime("09:18:22", "18:10:00") ➔ "08:51:38"  
etime("18:10:00", "09:18:22") ➔ "15:08:22"  
etime(time(), "17:30:45") ➔ "01:29:38"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

ftime(c1,c2)

Die Funktion **ftime**(FormatTime, formatiere Zeitstring) formatiert einen Zeitstring **c1** mit den in **c2** definierten Vorgaben.

Parameter:

c1 → ein gültiger Zeitstring in der Form "**hh:mm:ss**"

c2 → ein gültiger Zeitformatstring, siehe unten.

Der **Zeitformatstring** kann aus verschiedenen Teilen bestehen:

Stunden:

h → Stunde einstellig

hh → Stunde zweistellig

Minuten:

m → Minuten einstellig

mm → Minuten zweistellig

Sekunden:

s → Sekunden einstellig

ss → Sekunden zweistellig

Sonderzeichen:

: → Trennzeichen

'beliebige Zeichen' → innerhalb der Zeichen '' können Sie beliebige **tabbare** Zeichen (keine Ersatzzeichen, keine komplexen Funktionen), eingeben,

```
ftime("09:17:03","h:mm:ss")
```

```
➔ "9:17:03"
```

```
ftime("09:17:03","h' Uhr, 'm' Minuten, 's' Sekunden'")
```

```
➔ "9 Uhr, 17 Minuten, 3 Sekunden"
```

```
ftime("09:17:03","h:mm")
```

```
➔ "9:17"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

sec()

Die Funktion **sec**(seconds, Sekunden) gibt die Anzahl der Sekunden seit Mitternacht zurück.

Annahme: Die aktuelle Uhrzeit ist **09:15:15**

Parameter:

kein Parameter

```
sec()
```

➔ 33315

Die Rückgabe ist eine Zahl.

sec2time(n1)

Die Funktion **sec2time**(SecondsToTime, SekundenInZeit) rechnet die mit **n1** übergebene Anzahl von Sekunden in eine Zeitangabe der Form

"hh:mm:ss"

um.

Annahme: Die aktuelle Uhrzeit ist **09:15:15**

Parameter:

n1 → Sekunden

```
sec2time(33315)
```

➔ "09:15:15"

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

time2sec(c1)

Die Funktion time2sec(TimeToSeconds, ZeitInSekunden) rechnet eine übergebene Zeitangabe der Form

"hh:mm:ss"

in Sekunden um.

Parameter:

c1 → Zeitangabe "hh:mm:ss"

```
time2sec("09:15:15")
```

Die Rückgabe ist eine Zahl.

Bedienungsanleitung: SearARep

3.2.4.2.5 Textkettenmanipulationen (Stringmanipulationen)**atrim(c1,|c2|)**

Die Funktion **atrim**(AllTrim, alles abschneiden) löscht in **c1** alle Zeichenfolgen **c2**, die **direkt hintereinander** folgen und am **Anfang** oder am **Ende** des Strings stehen. Wenn Sie **c2** nicht angeben, setzt **SearARep** ein Leerzeichen ein.

Parameter:

- c1** → String, der beschnitten werden soll
c2 → die Zeichenfolge, die rechts und links abgeschnitten werden soll.
Wenn Sie **c2** nicht angeben, wird das Leerzeichen genommen.

```
atrim("      1234      ") → "1234"  
atrim("*****1234*****", "") → "1234"  
atrim("*****1234*****", "***") → "*1234*"  
atrim("abcaabc1234abcabc", "abc") → "aabc1234"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

ltrim(c1,|c2|)

Die Funktion **ltrim**(LeftTrim, links abschneiden) löscht in **c1** alle Zeichenfolgen **c2**, die **direkt hintereinander** folgen und am **Anfang** des Strings stehen. Wenn Sie **c2** nicht angeben, setzt **SearARep** ein Leerzeichen ein.

Parameter:

- c1** → String, der beschnitten werden soll
- c2** → die Zeichenfolge, die links abgeschnitten werden soll.
Wenn Sie **c2** nicht angeben, wird das Leerzeichen genommen.

```
ltrim("      1234      ") → "1234      "
ltrim("*****1234*****", "") → "1234*****"
ltrim("*****1234*****", "***") → "*1234*****"
ltrim("abcaabc1234abcabc", "abc") → "aabc1234abcabc"
```

Die Rückgabe ist ein String.

rtrim(c1,|c2|)

Die Funktion **rtrim**(RightTrim, rechts abschneiden) löscht in **c1** alle Zeichenfolgen **c2**, die **direkt hintereinander** folgen, und am **Ende** des Strings stehen. Wenn Sie **c2** nicht angeben, setzt **SearARep** ein Leerzeichen ein.

Parameter:

- c1** → String, der beschnitten werden soll
- c2** → die Zeichenfolge, die rechts abgeschnitten werden soll.
Wenn Sie **c2** nicht angeben, wird das Leerzeichen genommen.

```
atrim("      1234      ") → "      1234"
atrim("*****1234*****", "") → "*****1234"
atrim("*****1234*****", "***") → "*****1234*"
atrim("abcaabc1234abcaabc", "abc") → "abcaabc1234abca"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

fl(c1|n1,n2,c3)

Die Funktion **fl** (FillLeft) **formatiert** den String **c1** oder die Zahl **n1** (nach Umwandlung in einen String), indem sie den String **c3** so oft vor **c1** oder **n1** einfügt, bis der **Gesamtstring** die Länge **n2** erreicht hat:

Parameter:

c1 → String, der von links aufgefüllt werden soll.

oder

n1 → Zahl, die von links aufgefüllt werden soll.

n2 → Gesamtlänge des zu erzeugenden Strings

n2 → String, mit dem aufgefüllt werden soll.

<code>fl(34,10,"*")</code>	➔	<code>"*****34"</code>
<code>fl(343.17,20,"!*!")</code>	➔	<code>"!*!*!*!*!*!*!*!*343,17"</code>
<code>fl("Werner",20,"#")</code>	➔	<code>"#####Werner"</code>

Die Rückgabe ist ein String.

fr(c1|n1,n2,c3)

Die Funktion **fr** (FillRight) **formatiert** den String **c1** oder die Zahl **n1** (nach Umwandlung in einen String), indem sie den String **c3** so oft nach **c1** oder **n1** einfügt, bis der **Gesamtstring** die Länge **n2** erreicht hat:

Parameter:

c1 → String, der von rechts aufgefüllt werden soll.

oder

n1 → Zahl, die von rechts aufgefüllt werden soll.

n2 → Gesamtlänge des zu erzeugenden Strings

n2 → String, mit dem aufgefüllt werden soll.

<code>fl(34,10,"*")</code>	➔	<code>"34*****"</code>
<code>fl(343.17,20,"!*!")</code>	➔	<code>"343,17!*!*!*!*!*!*!"</code>
<code>fl("Werner",20,"#")</code>	➔	<code>"Werner#####"</code>

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

chr(n1,n2,n3,...,nn)

Die Funktion **chr**(Character, Zeichen) gibt die als Dezimalcodes eingegebenen Zeichen als Zeichenkette zurück. Beachten Sie bitte, dass das Ergebnis vom verwendeten Zeichensatz abhängig ist.

Parameter:

n1 ... nn → beliebig viele ganze Zahlen zwischen 0 und 255.

Für das Beispiel wird der **ANSI-Code** vorausgesetzt:

```
chr(80,101,114,112,108,105,101,115) → "Perplies"
```

Die Rückgabe ist ein String.

left(c1, n2)

Die Funktion **left**(links) schneidet **n2**-Zeichen von **c1** auf der linken Seite ab und gibt sie zurück.

Parameter:

c1 → String, von dem von **links** abgeschnitten werden soll.

n2 → Anzahl der Zeichen, die abgeschnitten werden sollen.

```
left("An diesem Freitag",9) → "An diesem"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

rightc1, n2)

Die Funktion **right**(rechts) schneidet **n2**-Zeichen von **c1** auf der rechten Seite ab und gibt sie zurück.

Parameter:

c1 → String, von dem von **rechts** abgeschnitten werden soll.

n2 → Anzahl der Zeichen, die abgeschnitten werden sollen.

```
left("An diesem Freitag",7)    ➔ "Freitag"
```

Die Rückgabe ist ein String.

substr(c1, n2, |n3|)

Die Funktion **substr**(Teilstring) entnimmt ab der **n2-ten** Stelle von **c1** **n3** Zeichen und gibt sie zurück.

Wenn Sie **n3** weglassen, werden alle Zeichen ab der **n2-ten** Stelle entnommen.

Parameter:

c1 → String, aus dem etwas herausgeschnitten werden soll.

n2 → Position, ab der etwas herausgeschnitten werden soll.

n3 → Die Anzahl der Zeichen, die herausgeschnitten werden sollen.
Wird dieser Parameter weggelassen, werden alle Zeichen bis zum Ende herausgeschnitten.

```
substr("An diesem Freitag",4,6) ➔ "diesem"
```

```
substr("An diesem Freitag",4)  ➔ "diesem Freitag"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

expand(|c1|,c2,n3)

Mit der Funktion **expand()** können Sie **innerhalb** einer bestimmten Textkette **ein Zeichen** oder **eine Zeichenkette suchen** und sie um einen **bestimmten Faktor expandieren**:

Parameter:

- c1** → ist die Textkette, in der **etwas gesucht** und **verändert** wird. Wenn Sie **c1** nicht angeben, wird **c1** automatisch durch die **gerade bearbeitete Datei** ersetzt.
- c2** → ist die Textkette, **nach der gesucht** wird und **die expandiert** wird. Dieser Parameter muss angegeben werden.
- n3** → ist **die Zahl** (ganze Zahl) **um die** die Zeichenkette expandiert werden soll. Dieser Parameter muss angegeben werden.

```
expand("abcabcadefgh", "ab", 3) ➔ "abababcabababcadefgh"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

red(|c1|,c2,n3)

Mit der Funktion **red**(reduce, reduzieren) können Sie **innerhalb** einer bestimmten Textkette **ein Zeichen** oder **eine Zeichenkette suchen** und sie auf eine bestimmte Anzahl reduzieren.

Parameter:

- c1** → ist die Textkette in der **etwas gesucht** und **verändert** wird. Wenn Sie **c1** nicht angeben, wird **c1** automatisch durch die **gerade bearbeitete Datei** ersetzt.
- c2** → ist die Textkette **nach der gesucht** wird und **die reduziert** wird. Dieser Parameter muss angegeben werden.
- n3** → ist **die Zahl** (ganze Zahl) **auf die** die Zeichenkette reduziert werden soll. Wird diese Zahl nicht angegeben, setzt **SearARep** automatisch die Zahl 1 ein.

```
red(" ")  
➔ reduziert in der aktuellen Datei Mehrfachleerzeichen  
auf ein Leerzeichen.
```

```
red("\kCRLF\",2)  
➔ reduziert in der aktuellen Datei mehrfache Zeilen-  
enden auf genau zwei Zeilenenden.
```

```
red("12232225552","2",2) -> "1223225552"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

convert(c1, c2)

Die Funktion **convert**(konvertieren) konvertiert den String **c1** mit den Konvertierregeln in der Konvertiertabelle **c2** und gibt die veränderte Textkette **c1** zurück.

Die Funktion `convert()` ermöglicht es Ihnen also, alle Regeln des Programmes **SearARep** auf jede beliebige (**Teil**)Textkette anzuwenden.

Parameter:

c1 → ist die Textkette die konvertiert werden soll

c2 → Ist der Name (+ vollständige Pfadangabe) der Konvertiertabelle

```
convert("War das alles?", "c:\Konvertiertabellen\t1.txt")
```

Das Ergebnis dieser Funktion hängt vom Inhalt der Konvertiertabelle ab und ist ein String.

Bedienungsanleitung: SearARep

ins(c1, c2, n3)

Die Funktion **ins**(insert, einfügen) fügt den String **c2** an der Position **n3** in in die Textkette **c1** ein und gibt die veränderte Textkette **c1** zurück.

Parameter:

- c1** → ist die Textkette, in die etwas eingefügt werden soll.
- c2** → ist die Textkette, die eingefügt werden soll.
- n1** → ist die Position, an der die Textkette eingefügt werden soll.

```
ins("War das alles?", "wirklich ",8)
➔ "War das wirklich alles?"
```

Die Rückgabe ist ein String.

del(c1, n2, n3)

Die Funktion **del**(delete, löschen) löscht in der Textkette **c1** an der Position **n2** die Anzahl von **n3** Zeichen.

Parameter:

- c1** → ist die Textkette, in der etwas gelöscht werden soll.
- c2** → ist die Position, an der etwas gelöscht werden soll.
- n1** → enthält die Anzahl der Zeichen, die gelöscht werden sollen.

```
del("War das wirklich alles?", 8,9)
➔ "War das alles?"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

slen(c1)

Die Funktion **slen**(StringLength, Stringlänge) gibt die Anzahl der Zeichen einer Textkette zurück.

Parameter:

c1 → ist die Textkette, deren Länge ermittelt werden soll.

```
slen("War das alles?") → 14
```

Die Rückgabe ist eine Zahl.

sr(c1,c2,c3)

Die Funktion **sr**(SearchandReplace, Textersetzung) sucht im String **c1** alle Vorkommen des Strings **c2** und ersetzt sie durch den String **c3**.

Parameter:

c1 → ist der String, in dem gesucht werden soll.

c2 → ist der String der gesucht werden soll.

c3 → ist der String mit dem ersetzt werden soll.

```
sr("Wer hat das gemacht", "Wer", "Emil") →  
"Emil hat das gemacht"
```

Die Rückgabe ist ein String.

Bedienungsanleitung: SearARep

3.2.4.2.6 Suchfunktionen

Mit den Suchfunktionen können Sie etwas suchen und in Verbindung mit den **Vergleichsfunktionen** und den **Bedingungsfunktionen** die **Ergebnisse auswerten** und in **Abhängigkeit dieser Ergebnisse** weitere **Aktionen starten**.

wisfl(c1,c2,n3)

Die Funktion **wisfl**(WhereIsFromLeft,WolstVonLinks) sucht den String **c1** innerhalb der Textkette **c2** und gibt seine Position zurück. Mit **n3** können Sie bestimmen, ab welcher Position gesucht werden soll.

Parameter:

c1 → Suchbegriff

c2 → String in dem gesucht werden soll

n3 → Position, ab der gesucht werden soll.

```
wisfl("Louise","im Wald ging Louise den ersten Weg rechts")  
➔ 14
```

```
wisfl("der","er war der Dritte, der den Ast aufhob")  
➔ 8
```

```
wisfl("der","er war der Dritte, der den Ast aufhob",9)  
➔ 20
```

Die zurückgegebene Position ist eine Zahl.

Bedienungsanleitung: SearARep

wisfr(c1,c2,n3)

Die Funktion **wisfr**(WhereIsFromRight,WolstVonRechts) sucht den String **c1** innerhalb der Textkette **c2** von rechts und gibt seine Position zurück. Mit **n3** können Sie bestimmen, ab welcher Position gesucht werden soll.

Parameter:

- c1** → Suchbegriff
- c2** → String in dem gesucht werden soll
- n3** → Position, ab der gesucht werden soll.

```
wisfl("der","er war der Dritte, der den Ast aufhob")  
➔ 20
```

```
wisfl("der","er war der Dritte, der den Ast aufhob",21)  
➔ 0
```

Die zurückgegebene Position ist eine Zahl.

instr(c1,c2)

Die Funktion **instr**(IsInString, IstImStringEnthalten) prüft, ob **c1** in **c2** enthalten ist.

Parameter:

- c1** → Suchbegriff
- c2** → String in dem gesucht werden soll

```
instr("der","er war der Dritte, der den Ast aufhob")  
➔ wahr!
```

```
instr("die","er war der Dritte, der den Ast aufhob")  
➔ nicht wahr!
```

Der Rückgabewert ist **logisch** (wahr, nicht wahr), siehe **Vergleichsfunktionen** auf Seite 72 und die Funktion **if()/ifn()** auf Seite 77.

Bedienungsanleitung: SearARep

3.2.4.2.7 Vergleichsfunktionen

Mit den Vergleichsfunktionen können Sie den Inhalt **gleicher** Datentypen **vergleichen** und mit der **if()**-Funktion weitere **Aktionen in Abhängigkeit vom Ergebnis** vornehmen.

bt(u1,u2,u3)

Mit der Funktion **bt**(between, zwischen) können Sie prüfen, ob der Wert **u1** zwischen **u2** und **u3** liegt.

Parameter:

- u1** → beliebige Zahl oder Textkette, zu testender Wert
- u2** → beliebige Zahl oder Textkette, unterer Wert
- u3** → beliebige Zahl oder Textkette, oberer Wert

Alle drei Parameter müssen vom gleichen Datentyp sein!

<code>bt(3,2,10)</code>	➔ wahr
<code>bt(11,2,10)</code>	➔ nicht wahr
<code>bt("berta","adam","eva")</code>	➔ wahr
<code>bt("werner","adam","eva")</code>	➔ nicht wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

Bedienungsanleitung: SearARep

eq(u1,u2)

Mit der Funktion **eq**(equal,gleich) können Sie prüfen, ob der Wert **u1** gleich dem Wert **u2** ist.

Parameter:

u1 → beliebige Zahl, Textkette oder logischer Wert

u2 → beliebige Zahl, Textkette oder logischer Wert

Beide Parameter müssen vom gleichen Datentyp sein!

eq(3,3)	➔ wahr
eq(11,2)	➔ nicht wahr
eq("berta","adam")	➔ nicht wahr
*eq("werner","wer")	➔ wahr
*eq("wer","werner")	➔ nicht wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

* Beachten Sie für diesen Sonderfall bitte auch die Funktion **eeq()**

eeq(u1,u2)

Mit der Funktion **eeq**(EqualEqual,genau gleich) können Sie prüfen, ob der Wert **u1** genau gleich dem Wert **u2** ist.

Parameter:

u1 → beliebige Zahl, Textkette oder logischer Wert

u2 → beliebige Zahl, Textkette oder logischer Wert

Beide Parameter müssen vom gleichen Datentyp sein!

eeq(3,3)	➔ wahr
eeq(11,2)	➔ nicht wahr
eeq("berta","adam")	➔ nicht wahr
eeq("werner","wer")	➔ nicht wahr
eeq("wer","werner")	➔ nicht wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

Bedienungsanleitung: SearARep

noteq(u1,u2)

Mit der Funktion **noteq**(NotEqual,ungleich) können Sie prüfen, ob der Wert **u1** ungleich dem Wert **u2** ist.

Parameter:

u1 → beliebige Zahl, Textkette oder logischer Wert

u2 → beliebige Zahl, Textkette oder logischer Wert

Beide Parameter müssen vom gleichen Datentyp sein!

<code>noteq(3,3)</code>	➔ nicht wahr
<code>noteq(11,2)</code>	➔ wahr
<code>noteq("berta","adam")</code>	➔ wahr
<code>*noteq("werner","wer")</code>	➔ nicht wahr
<code>*noteq("wer","werner")</code>	➔ wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

* Beachten Sie für diesen Sonderfall bitte auch die Funktion **eeq()**

gt(u1,u2)

Mit der Funktion **gt**(greater,größer) können Sie prüfen, ob der Wert **u1** größer dem Wert **u2** ist.

Parameter:

u1 → beliebige Zahl, Textkette oder logischer Wert

u2 → beliebige Zahl, Textkette oder logischer Wert

Beide Parameter müssen vom gleichen Datentyp sein!

<code>gt(3,3)</code>	➔ nicht wahr
<code>gt(11,2)</code>	➔ wahr
<code>gt("berta","adam")</code>	➔ wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

Bedienungsanleitung: SearARep

gteq(u1,u2)

Mit der Funktion **gteq**(GreaterEqual, größer gleich) können Sie prüfen, ob der Wert **u1** größer gleich dem Wert **u2** ist.

Parameter:

u1 → beliebige Zahl, Textkette oder logischer Wert

u2 → beliebige Zahl, Textkette oder logischer Wert

Beide Parameter müssen vom gleichen Datentyp sein!

eq(3,3)	➔ wahr
eq(11,2)	➔ wahr
eq("berta","adam")	➔ wahr
eq("adam","berta")	➔ nicht wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

ls(u1,u2)

Mit der Funktion **ls**(less, kleiner) können Sie prüfen, ob der Wert **u1** kleiner dem Wert **u2** ist.

Parameter:

u1 → beliebige Zahl, Textkette oder logischer Wert

u2 → beliebige Zahl, Textkette oder logischer Wert

Beide Parameter müssen vom gleichen Datentyp sein!

ls(3,3)	➔ nicht wahr
ls(11,2)	➔ nicht wahr
ls("berta","adam")	➔ nicht wahr
ls("adam","berta")	➔ wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

Bedienungsanleitung: SearARep

Iseq(u1,u2)

Mit der Funktion **Iseq**(LessEqual, kleiner gleich) können Sie prüfen, ob der Wert **u1** kleiner gleich dem Wert **u2** ist.

Parameter:

u1 → beliebige Zahl, Textkette oder logischer Wert

u2 → beliebige Zahl, Textkette oder logischer Wert

Beide Parameter müssen vom gleichen Datentyp sein!

<code>Iseq(3,3)</code>	➔ wahr
<code>Iseq(11,2)</code>	➔ nicht wahr
<code>Iseq("berta","adam")</code>	➔ nicht wahr
<code>Iseq("adam","berta")</code>	➔ wahr

Der Rückgabewert ist logisch (wahr/nicht wahr)

Bedienungsanleitung: SearARep

3.2.4.2.8 Bedingungsfunktionen

if(I1,u1,u2)

Mit der Funktion **if**(wenn, dann) können Sie bestimmen, dass ein Wert in Abhängigkeit des logischen Wertes **I1** zurückgegeben wird.

Parameter:

- I1** → logischer Wert wahr, nicht wahr, Rückgaben von Vergleichs- und Suchfunktionen.
u1 → beliebiger Datentyp → Rückgabe bei **wahrer** Bedingung.
u2 → beliebiger Datentyp → Rückgabe bei **nicht wahrer** Bedingung.

```
if(wahr, "willi", "otto")           ➔ "willi"
if(nicht wahr, "willi", "otto")    ➔ "otto"
if(bt(3,2,6), "willi", "otto")     ➔ "willi"
if(instr("a", "anton"), "anton", "otto") ➔ "anton"
if(instr("a", "otto"), "anton", "otto") ➔ "otto"
```

Der Rückgabewert hängt von den **Datentypen u1** und **u2** ab

ifn(I1,u1,u2)

Mit der Funktion **ifn**(wenn nicht, dann) können Sie bestimmen, dass ein Wert in Abhängigkeit des logischen Wertes **I1** zurückgegeben wird.

Parameter:

- I1** → logischer Wert wahr, nicht wahr, Rückgaben von Vergleichs- und Suchfunktionen.
u1 → beliebiger Datentyp → Rückgabe bei **wahrer** Bedingung.
u2 → beliebiger Datentyp → Rückgabe bei **nicht wahrer** Bedingung.

Sowohl beim if- als auch beim ifn-Kommando wird nur der Parameter ausgewertet, der wahr ist. Bei umfangreichen Bearbeitungen hängt die Verarbeitungsgeschwindigkeit entscheidend von der Häufigkeit des Eintreffens der Bedingung und der Komplexität des benutzten Parameters ab.

Der Rückgabewert hängt von den **Datentypen u1** und **u2** ab

Optimieren Sie den Ablauf durch die **richtige Auswahl** der Bedingung.

Bedienungsanleitung: SearARep

3.2.4.2.9 Dateifunktionen

Die Dateifunktionen dienen dem **Einlesen**, **Speichern** und **Bearbeiten** von Dateien über die in der Dateiauswahl von **SearARep** hinaus gewählten Dateien.

So können Sie mit diesen Funktionen beliebige Dateien in Textketten einfügen oder auch aus den gerade bearbeiteten Dateien neue Dateien erzeugen oder Teile extrahieren.

rf(c1)

Die Funktion **rf**(ReadFile, LeseDateiEin) liest die in **c1** angegebene Datei ein und gibt ihren **Inhalt** zurück.

Parameter:

c1 → Dateinamen mit vollständigem Verzeichnispfad

```
rf("c:\Bausteine\Artikel-1345.txt")  
➔ Inhalt der Datei Artikel-1345.txt
```

Der Rückgabewert ist ein String.

sf(c1,c2)

Die Funktion **sf**(StoreFile, SpeichereDatei) speichert die in **c1** angegebene Textkette auf dem unter **c2** angegebenen Platz und Namen ab.

Achtung: Eine eventuell **vorhandene Datei** wird **überschrieben!**

Parameter:

c1 → zu speichernde Textkette

c2 → der vollständige Dateiname

```
sf("aaaa", "c:\Bausteine\Artikel-1345.txt")➔
```

Sollte der Dateiname **keine Pfadangabe** enthalten, wird als Standardpfad das **Ausgabeverzeichnis** der bearbeiteten Datei eingesetzt.

Die Funktion gibt die abgespeicherte Textkette zurück.

Bedienungsanleitung: SearARep

cofn(c1)

Die Funktion **cofn**(ChangeOutputFileName, ÄndereAusgabeDateiNamen) ersetzt den **ursprünglichen Dateinamen** durch **c1**.

Parameter:

c1 → neuer (gültiger) Dateiname

Annahme:

Der ursprüngliche Name lautet: "Bedienungsanleitung.txt"

```
°1 !cofn(print($C_ $OutputName, ".html")) °3 →
"Bedienungsanleitung.html"
```

Die Funktion gibt den neuen Dateinamen zurück.

Normalerweise sollten Sie diesen Befehl im **Feld 2** mit dem **Prefix "!"** verwenden.

lct(c1)

Die Funktion **lct**(LoadCommandTable, LadeKonvertiertTabelle) lädt die Konvertiertabelle **c1** in ein Array.

Parameter:

c1 → neuer (gültiger) Dateiname

```
°1 !sv("$A_Rohtext", lct("c:\KnvTabellen\Rohtext.knv")) °3 →
Array mit der Konvertiertabelle Rohtext als Inhalt.
```

Die Funktion lct gibt ein Array zurück.

Verwenden Sie diesen Befehl wie im Beispiel in Verbindung mit der Funktion **sv** (Variable speichern) im **Feld 2** mit dem **Prefix "!"**, um wiederholt zu verwendende Tabellen nur einmal von der Festplatte einlesen zu müssen.

Bedienungsanleitung: SearARep

3.2.4.2.10 Variablenfunktionen

Variablen sind Speicherplätze innerhalb des **SearARep**-Programmes, mit deren Hilfe Sie sich Informationen vorübergehend merken können und auf die Sie bei Bedarf wieder zugreifen können.

Über diese **selbst definierten** Variablen hinaus, bietet Ihnen **SearARep** auch noch einige **interne Systemvariablen**, mit denen Ihnen das Programm Informationen zur Verfügung stellt, die sich aus dem Programmablauf ergeben.

Syntax der Variablen:

Für die Namen der Variablen gelten die **folgenden Vorgaben**:

1. Ein Variablenname muss mit dem \$-Zeichen beginnen.
2. Das 2. Zeichen beschreibt den Datentyp: C/N/I/F/L/D/A/\$, siehe unten
3. Das 3. Zeichen ist das _-Zeichen
4. Ist das 4. Zeichen ein \$-Zeichen, handelt es sich um eine Systemvariable.
5. Die weiteren Zeichen sind beliebig.

Systemvariablen:

- \$C_\$Input** → enthält den **Inhalt** der gerade **bearbeiteten Datei**
- \$C_\$\$StartInputPath** → enthält das Ausgangsverzeichnis der ausgewählten Dateien. Bei **Verzeichnisstrukturen** das **oberste Verzeichnis**.
- \$C_\$InputPath** → enthält das **Verzeichnis** der **aktuell bearbeiteten Datei**.
- \$C_\$InputNameWExt** → enthält den Namen der **aktuell bearbeiteten Datei** einschließlich der Dateinamenserweiterung.
- \$C_\$InputName** → enthält den Namen der **aktuell bearbeiteten Datei** ohne die Dateinamenserweiterung.
- \$C_\$\$OutputNameWExt** → enthält den Namen der **aktuellen Ausgabedatei** einschließlich der Dateinamenserweiterung.
- \$C_\$\$OutputName** → enthält den Namen der **aktuellen Ausgabedatei** ohne die Dateinamenserweiterung.

Bedienungsanleitung: SearARep

- \$C_ \$OutputFileName** → enthält den kompletten Namen der **aktuellen Ausgabedatei** inklusive Pfad und Erweiterung.
- \$C_ \$LastWildcard** → enthält die Textkette, die mit der **letzten Wildcard(Joker)-Funktion** gefunden wurde.
- \$C_ \$LastFigWildcard** → enthält die Textkette, die mit der **letzten Zahlenwildcard(Joker)-Funktion** gefunden wurde.
- \$I_ \$LastIPCPosition** → enthält die **Position**, an der der **letzte Suchbegriff innerhalb** einer **IPC-Konvertierregel** gefunden wurde.
- \$C_ \$ExePath** → enthält das **Programmverzeichnis** von **SearARep**

Datentypen von Variablen:

Die zweite Stelle im Variablennamen bestimmt den **Datentyp der Variablen**:

- C** → Textkette
- N** → Zahl, numerischer Ausdruck
- I** → Zahl, ganze positive/negative Zahl
- F** → Zahl, positive/negative Gleitkommazahl
- L** → logischer Ausdruck, Rückgabe einer Vergleichs- oder Suchfunktion.
- D** → Datum
- A** → Array mit beliebigen Elementen
- \$** → Makro, siehe Makroverarbeitung weiter unten.

Bedienungsanleitung: SearARep

sv(c1,u2)

Die Funktion **sv**(StoreVariable, SpeichereVariable) speichert **u2** unter dem Namen **c1** in einem internen Speicher des SearARep-Systems ab.

Wenn die Variable **c1** noch **nicht vorhanden war**, wird der Speicherplatz **neu eingerichtet**.

War die Variable mit dem Namen **c1** schon **vorhanden**, wird der Inhalt des Speicherplatzes **überschrieben**.

Beachten Sie bitte die **richtige Verwendung der Datentypen**.

Parameter:

c1 → Name der Variablen als Textkette

u2 → beliebiger Datentyp

```
sv("$C_Vorname", "Werner")
➔ speichert "Werner" unter dem Namen $C_Vorname ab.
sv("$I_Zähler", 0)
➔ speichert 0 unter dem Namen $I_Zähler ab.
sv("$N_Zähler", add($N_Zähler, 1))
➔ speichert die Summe $N_Zähler+1 unter $N_Zähler ab.
sv("$$Zähler", 'sv("$N_Zähler", add($N_Zähler, 1))')
➔ speichert das Makro sv("$N_Zähler", add($N_Zähler, 1))
  $$Zähler
➔ erhöht die Variable $N_Zähler um 1
print($N_Zähler)
➔ gibt den Inhalt der Variable $N_Zähler als String aus.
```

Die Funktion gibt den gespeicherten Wert als String zurück. Zuvor findet eine Datentypumwandlung statt:

Datentyp:

String	→ keine Umwandlung
Numerisch(N/I/F)	→ Wert als String
Logisch	→ "wahr"/"falsch"
Makro	→ Wert als String
Array	→ "Array"

Bedienungsanleitung: SearARep

dv(c1)

Die Funktion **dv**(DeleteVariable, LöscheVariable) löscht **Namen** und **Inhalt** der Variablen **c1**.

Die Anzahl der benutzten Variablen **beeinflusst die Verarbeitungsgeschwindigkeit** des Programmes **SearARep**.

Der Inhalt der benutzten Variablen **erhöht den Speicherbedarf** des Programmes **SearARep**.

Löschen Sie deshalb nicht mehr benötigte Variablen.

Mit Beendigung des Programmes SearARep werden alle Variablen gelöscht.

Parameter:

c1 → der Variablenname

```
dv("$N_Zähler") →
```

Diese Funktion gibt den letzten Wert der Variablen zurück. Eine Makrovariable wird nicht mehr ausgeführt.

iev(c1)

Die Funktion **iev**(IfExistVariable), existiert die Variable) ermittelt, ob eine Variable bereits definiert wurde.

Parameter:

c1 → Variablenname

```
sv("$C_Name", "a")  
iev("$C_Name") → logisch: wahr
```

Die Funktion gibt einen logischen Wert zurück.

Bedienungsanleitung: SearARep

Inhalt einer Variablen auslesen

Lesen Sie den **Inhalt einer Variablen** aus, indem Sie den **Variablennamen ohne Begrenzungszeichen** an Stelle einer **komplexen Funktion** angeben.

Variablen, die Makros enthalten, werden dann automatisch ausgeführt.

Makroverarbeitung

Sie können **alle Aufrufe** von komplexen Funktionen **einzel**n oder **verschachtelt** als Textkette in einer Variablen mit dem **Datentyp Makro** abspeichern.

Wenn Sie innerhalb der Textkette des Befehls das "-"Zeichen verwenden müssen, benutzen Sie bitte zur Kennzeichnung der Befehltextkette das '-'Zeichen.

Achtung:

Die Verwendung von Makros kann die Verarbeitungsgeschwindigkeit von SearARep erheblich herabsetzen.

Bedienungsanleitung: SearARep

3.3 **Verwendung von mehreren Konvertiertabellen**

Konvertiertabellen können im Rahmen der Speichergröße des verwendeten PC's beliebig groß sein und eine beliebige Anzahl von Einträgen haben.

Dennoch kann es unter bestimmten Umständen sinnvoll sein, **mehrere** Konvertiertabellen zu verwenden.

3.3.1 **Aufgabenbezogene Konvertiertabellen**

Unter **aufgabenbezogenen** Konvertiertabellen verstehe ich solche Tabellen, die **bestimmte Aufgaben** erledigen, z. B.:

Rohtextaufbereitung

Korrektur häufig vorkommender Rechtschreibungsfehler.

Einfügen bestimmter Abstände

Normierung von Schreibweisen

Auszeichnung von Texten mit Befehlen

3.3.2 **Auftragsbezogene Konvertiertabellen**

Auftragsbezogene Konvertiertabellen benötigen oft **nur einen Teil** der **aufgabenbezogenen** Konvertiertabellen und unterscheiden sich naturgemäß von Auftrag zu Auftrag.

Sie könnten jetzt aus den einzelnen **aufgabenbezogenen** Tabellen eine gemeinsame **auftragsbezogene** Tabelle erzeugen und diese unter dem Auftragsnamen ablegen.

Ein **entscheidender Nachteil** dieses Verfahrens ist es, dass Sie mit dieser Vorgehensweise **viele Kopien** erzeugen. Stellt sich zu irgendeinem Zeitpunkt heraus, dass in einer der Teiltabellen ein Fehler war, so müssen Sie in **allen erzeugten Kopien** diesen **Fehler beheben**.

Verwenden Sie deshalb besser die Möglichkeit, in einer Konvertiertabelle weitere Tabellen aufzurufen.

Da **darüber hinaus** auch jede Teiltabelle wieder aus mehreren Tabellen bestehen kann, nenne ich diese Art von Tabellenaufruf den **geschachtelten Aufruf** von Konvertiertabellen.

Bedienungsanleitung: SearARep

3.3.3 Geschachtelte Konvertiertabellen

Innerhalb einer Konvertiertabelle können Sie **beliebig oft** eine weitere Konvertiertabelle aufrufen. Dies geschieht durch das Kommando "include" im **Feld 1** einer Konvertierregel und durch die Eingabe des vollständigen Dateinamens der gewünschten Tabelle in **Feld 3**.

Beispiel:

```
// Beispiel für geschachtelte Konvertiertabellen:  
// diese Tabelle korrigiert die 500 häufigsten Lesefehler der Vorlagen des Projekts-17  
include c:\Konvertiertabellen\OCR-Projekt-17.knv  
// diese Tabelle beseitigt "harte Zeilenschaltungen", Trennungen,  
// leere Absätze, Mehrfachleerzeichen  
include c:\Konvertiertabellen\Rohtext.knv  
// diese Tabelle wandelt wandelt "-Zeichen in deutsche Anführungszeichen um  
include c:\Konvertiertabellen\Deutsche_Anführung.knv  
// diese Tabelle korrigiert häufig vorkommende Rechtschreibfehler juristischer Texte  
include c:\Konvertiertabellen\Korr_jur.knv  
// diese Tabelle normiert Abstände  
include c:\Konvertiertabellen\Abstand.knv  
// diese Tabelle normiert die Schreibweisen von Abkürzungen für juristische Texte  
// und enthält den Aufruf von zwei weiteren Tabellen für allgemeine Abkürzungen  
// und auch juristische Abkürzungen  
include c:\Konvertiertabellen\Abkürz_jur.knv  
// Diese Tabelle fügt Tags (Befehle) für Quark XPress ein  
include c:\Konvertiertabellen\Xpress-Tags(projekt-17).knv
```

Abb. 42 Beispiel einer geschachtelten Konvertiertabelle

Grundsätzlich kann **jede aufgerufene** Tabelle **wieder beliebig viele** Tabellen aufrufen, aber achten Sie bitte **unbedingt** darauf, dass eine **aufgerufene** Tabelle nicht die **aufrufende** Tabelle enthält.

Dies klingt reichlich kompliziert, führt aber, versehentlich angewendet, zu einer *Endlosschleife*. **SearARep** erkennt dies aber als Fehler und verweigert den Aufruf solch einer Tabelle.

Bedienungsanleitung: SearARep

3.4 IPC-Kommandos (Inline Programming Command)

Den **IPC-Kommandos** kommen bei der Arbeit mit **SearARep** eine besondere Bedeutung zu, indem Sie die Verarbeitung von komplexen Funktionen innerhalb des Ersatzbegriffes auslösen.

Es werden dabei die Kommandos **ipc** und **zipc** unterschieden. Diese beiden Kommandos gibt es wiederum in einer **normalen** und in einer **Plus-Version** (**ipc+**/**zipc+**)

Das **Verständnis** der **unterschiedlichen Wirkungen** dieser einzelnen Kommandos ist **äußerst wichtig** für den richtigen Einsatz der *komplexen Funktionen*.

Die **verschiedenen Konvertierregeln** werden intern auf **völlig unterschiedliche** Art abgearbeitet:

3.4.1 Einfache Konvertierregel

Bei der **einfachen Konvertierregel** werden alle Suchen- und Ersetzoperationen in **einem Arbeitsgang** über den zur Verfügung gestellten **Datenbereich** abgearbeitet, **ohne** dass auf die **einzelnen Ersetzungen Einfluss** genommen werden kann.

Dies ermöglicht eine extrem schnelle Arbeitsweise, verbietet aber eine Interpretation von komplexen Funktionen innerhalb des Ersetzungsvorganges. Es verhindert darüber hinaus auch Statusanzeigen auf dem Bildschirm.

Wenn Sie für eine **einfache Konvertierregel** das **IPC-Kommando** verwenden, so werden **komplexe Funktionen einmalig** je Konvertierregel **vor** dem eigentlichen **Ersetzungsvorgang** interpretiert und der neu erhaltene Ersetzungsstring wird bei **allen** Ersetzungen verwendet.

Sie können also **innerhalb** dieses Verarbeitungsschrittes **keine Informationen sammeln, Veränderungen vornehmen** oder den **Vorgang abbrechen**.

Sie können diese Art der Verarbeitung dadurch verändern, indem Sie die **Plus-Version** des IPC-Kommandos verwenden:

Bei der Eingabe von **ipc+** (**zipc+**) wird der Verarbeitungsvorgang **auf-**

Bedienungsanleitung: SearARep

gebrochen, das Programm sammelt gewisse Statusmeldungen und interpretiert vor jedem Ersetzungsvorgang eventuell vorhandene *komplexe Funktionen*. Die Aktivierung der **IPC-Plus-Funktion** reduziert die Verarbeitungsgeschwindigkeit erheblich.

Vor Einsatz dieser Funktion sollten Sie deshalb prüfen, ob die gestellte Aufgabe nicht auch mit einfachen Konvertierregeln lösbar wäre.

3.4.2 Wildcard- oder Jokerfunktionen

Die Wildcard- oder Jokerfunktion arbeitet grundsätzlich ähnlich wie die oben beschriebene **IPC-Plus-Funktion**.

Zuerst sucht das Programm den Suchbegriff **vor** dem Joker, merkt sich die Position, sucht dann den Begriff **nach** dem Joker und merkt sich auch diese Position.

Dann zerlegt das Programm den Datenbereich in **drei** Teile, den Teil **bis** zum Suchbegriff, den Teil **zwischen** den Suchbegriffen und den Teil **nach** den Suchbegriffen. Den Teil **zwischen** den Suchbegriffen stellt Ihnen das Programm in der Systemvariablen **\$C_ \$LastWildCard** zur Bearbeitung mit **komplexen Funktionen** zur Verfügung.

Anschließend wird die eigentliche Ersetzung vorgenommen.

Im nächsten Vorgang wird der Bereich **nach** der vorgenommenen Ersetzung durchsucht und weiter verarbeitet.

Auch **ohne** eingeschaltete **IPC-Funktion** ist diese Art der Verarbeitung **langsamer** als die Verarbeitung mit einer **einfachen Konvertierregel**.

Nach Einschalten der **IPC-Funktion reduziert** sich die Verarbeitungsgeschwindigkeit weiter, weil **vor jeder** vorgenommenen **Ersetzung** eine eventuell **vorhandene komplexe Funktion interpretiert** und **ausgeführt** wird.

Bedienungsanleitung: SearARep

3.4.3 Ziffernjoker

Die Verarbeitung der **Ziffernjoker** ist eine Mischung aus **beiden** zuvor beschriebenen Verfahren:

Jede Zahl des **definierten Zahlenbereichs** löst **einen** Vorgang für den gesamten Arbeitsbereich aus, das heißt, **alle** Ersetzungen werden für die **gleiche Zahl gemeinsam** vorgenommen.

Ein Nebeneffekt dieser Verarbeitungsstrategie besteht darin, dass Zahlen mehrfach ersetzt werden können,

Sie sollten deshalb darauf achten, dass zu suchende Zahlen im Suchstring möglichst durch andere Zeichen eingerahmt werden.

Die **Aktivierung** des **IPC-Kommandos** erlaubt Ihnen die **Ersetzung für eine Zahl** (**einen** Ersetzungsvorgang) mit *komplexen Funktionen* zu beeinflussen. Außerdem liefert Ihnen das Programm gewisse Statusmeldungen und die ersetzten Ziffern in einer Systemvariablen.

Bedienungsanleitung: SearARep

3.5 Allgemeine Hinweise zu Konvertiertabellen und Konvertierregeln

Mit den aufgeführten Konvertierregeln können Sie vermutlich über 90% Ihrer

Suchen-und-Ersetzen-Aufgaben

lösen.

Von den verbleibenden 10% sind manche Aufgaben durch die Verwendung **mehrerer Konvertierregeln nacheinander** lösbar.

Für den Fall, dass Sie einmal keine Lösung finden, sollten Sie mir Ihr Problem in einer Mail mitteilen. Verwenden Sie im Betreff bitte das Stichwort **Problem:SearARep**.

Wenn es doch eine Lösung gibt, werde ich sie Ihnen mitteilen.

Wenn es die Lösung nicht gibt, es sich aber um ein grundsätzliches Problem handelt, könnte es eventuell auch in einer neuen Version von **SearARep** gelöst werden.

Wenn das Problem für Sie **besonders wichtig** und **lösbar ist**, könnten Sie mich auch mit der Programmierung dieses Problem es beauftragen.

Fragen Sie einfach danach.

Schreiben Sie bitte an

*werner.perplies@weepee.de,
Betreff: SearARep*

Bedienungsanleitung: SearARep

4 Protokolldateien

Mit jedem Aufruf von **SearARep** legt das Programm im Verzeichnis

... \Programmverzeichnis \log

eine Protokolldatei mit dem Namen *000_SearARep.log* an. Sofern schon eine ältere Datei da ist, wird die dreistellige Ziffer hochgezählt.

Diese Protokolldatei informiert Sie im Normalfall über:

- den Zeitpunkt des Programmstartes
- das angelegte Ausgangsverzeichnis
- den Dateinamen der bearbeiteten Datei
- den Dateinamen der verwendeten Konvertiertabelle
- die benötigte Zeit für eine Datei
- den Zeitpunkt des Programmendes und der Programmlaufzeit.

Diese **normalen** Einträge sind durch

+++ Info:

gekennzeichnet.

Sollte SearARep einen Fehler erkennen, so wird dieser Eintrag durch

***** Fehler in ...**

gekennzeichnet.

Dabei versucht **SearARep** Ihnen eine möglichst genaue Information über den Fehler zu geben und teilt außerdem mit, in welchem Programmteil der Fehler aufgetreten ist.

Wenn der aufgetretene Fehler nicht durch Sie behoben werden kann, sollten Sie mir diese Informationen zuschicken.

Löschen Sie bitte die nicht mehr benötigten Protokolldateien regelmäßig.

Bedienungsanleitung: SearARep

5 Fehler

Ein Programm mit der Funktionsvielfalt von **SearARep** bietet leider auch eine Vielzahl von Fehlermöglichkeiten. Dabei sind völlig verschiedene Fehlerursachen möglich.

Es gibt Fehler, die Sie als Anwender verursachen können, und, hoffentlich sehr selten, Fehler, die ich als Programmierer zu vertreten habe:

5.1 Fehlermöglichkeiten (Anwendungsfehler)

5.1.1 Fehler, die keine Fehlermeldung auslösen

Diese Art von Fehlern kommt nach meiner Erfahrung in der Anfangszeit am häufigsten vor (**ohne Anspruch auf Vollständigkeit!**):

Symptom:

Alle Dateien wurden verarbeitet, die erzeugten Dateien enthalten keine oder zu wenig Änderungen.

Ursache:

Falsche Konvertiertabelle, **richtige** Konvertiertabelle mit **teilweise** oder **komplett auskommentierten** Konvertiereregeln.

Symptom:

falsche Ersetzungen, **einzelne, nicht ausgeführte** Ersetzungen

Ursache:

Fehler in den Konvertiereregeln, überprüfen Sie die entsprechenden Regeln

Symptom:

Die erzeugten Dateien sind leer, oder enthalten komplett ungewollten Text:

Ursache:

Fehler in der **Anwendung der Jokerfunktion**, **falsch** angewendete *komplexe Funktion* in Feld 2 (fehlendes "!")

Bedienungsanleitung: SearARep

Symptom:

Die erzeugten Dateien enthalten komplexe Funktionen:

Ursache:

Sie haben die äußere Kennung "**|\$C Funktion() |**" und oder das **IPC-Kommando** in Feld 1 vergessen.

Symptom:

Die Ersetzungen werden nicht in der erwarteten Art und Weise vorgenommen.

Ursache:

Alle Befehle einer Tabelle werden von **oben** nach **unten** abgearbeitet. Bereits **ausgeführte Ersetzungen** können die Ausführung einzelner Regeln verhindern. Ändern Sie ggf. die Reihenfolge Ihrer Konvertiereregeln.

Alle **aufgezählten Fehler** sind **keine Fehler** im Sinne des Programmes **SearARep**, denn Sie haben für das Programm durchaus richtige Anweisungen gegeben, aber es waren **falsche Anweisungen** im **Sinne des gewünschten Ergebnisses**.

5.1.2 Fehler, die zu Fehlermeldungen führen

Fast **kein Fehler**, der von **SearARep** erkannt wird, führt zu einem **Programmabbruch**. **Erkannte** Fehler werden gezählt und es wird eine Fehlermeldung in die Protokolldatei geschrieben.

Nach **Programmablauf** und **gemeldeten Fehlern**, können Sie die Fehlermeldung analysieren und die Fehler beheben.

Systembedingt kann die **Fehleranzahl** in manchen Fällen **recht groß** sein. Erschrecken Sie bitte nicht. Oft hat hier eine kleine Ursache eine Riesenwirkung.

Viele Konvertierungsvorgänge laufen bei **SearARep** intern als Programmschleife ab. Ein **einfacher Schreibfehler** in einem **wiederholt aufgerufenen Befehl** wird, da das Programm ja **bewusst nicht abgebrochen** wird, bei **jedem** Schleifendurchlauf **erneut erkannt, gezählt** und **ausgegeben**.

Manche Fehler sind auch eine **Folge** eines **vorausgegangenen Fehlers**.

Bedienungsanleitung: SearARep

So kommen bei einer großen Datei und und der Verwendung von komplexen Funktionen schnell einige tausend Fehler zusammen.

SearARep teilt Ihnen im Protokoll in der Regel folgende Informationen mit:

In **welcher Konvertiertabelle** ist der Fehler aufgetreten?

In **welcher Zeile** der Konvertiertabelle ist der Fehler aufgetreten?

Bei **welchem Kommando** ist der Fehler aufgetreten?

Welche Art von Fehler ist aufgetreten?

Manchmal wird der **fehlerhafte Teil** des Kommandos ausgegeben.

Zusätzlich werden **programminterne Informationen** ausgegeben.

Bedienungsanleitung: SearARep

5.1.3 Strategien zur Fehlersuche

Wenn der aufgetretene Fehler nicht so offensichtlich ist, dass er auf den ersten Blick erkannt wird, können Ihnen vielleicht die folgenden Hinweise helfen:

Welcher Fehler ist aufgetreten?

Fehler im Format der Konvertierregel:

Stimmen die **Feldbezeichnungen**?

°1/°2/°3/°4/°5, achten Sie besonders darauf, dass kein Zeichen (auch kein Leerzeichen oder Tabulator) zwischen dem Gradzeichen und der Zahl ist.

Stehen die **richtigen Kommandos** im **richtigen Feld**?

Dateifehler:

Die eingestellte **Konvertiertabelle** wurde **gelöscht** oder **verschoben**.

Die eingestellte **Konvertiertabelle** ist mit einem **ungeeigneten Editor** (noch) geöffnet.

Ausgewählte Dateien sind (noch) **geöffnet**, wurden während der Verarbeitung **gelöscht** oder **verschoben**.

Der **erzeugte Ausgabepfad** wurde für das Betriebssystem **zu lang**. Löschen Sie nicht mehr benötigte Ausgabeverzeichnisse von **SearA-Rep** oder benennen Sie diese Verzeichnisse um.

Fehler in komplexen Funktionen:

Sie haben die äußere hintere Klammer der Funktion vergessen ("|&C Funktion()>>> |"). Korrigieren Sie diesen Fehler!

Klammerfehler:

Die häufigste Fehlerursache dürften falsch gesetzte Klammern sein. Überprüfen Sie deshalb die Klammersetzung. Jeder geöffneten Klammer muss außerhalb von Textketten eine geschlossene Klammer gegenüberstehen. Viele Texteditoren bieten Ihnen eine spezielle Klammerprüffunktion (**Textpad** Strg+M) .

Bedienungsanleitung: SearARep

Wenn Ihnen eine solche Funktion nicht zur Verfügung steht, sollten Sie die Klammern der einzelnen Kommandos von **innen** nach **außen** prüfen.

Schneiden Sie dafür die einzelnen Kommandos von **innen** nach **außen** heraus und prüfen Sie jedes einzelne Kommando.

Nutzen Sie bei **tief verschachtelten Funktionen** die Möglichkeiten der **verbundenen Zeilen**. Mit ihrer Hilfe können Sie die **Klammern** der einzelnen Kommandos **so einrücken**, dass die zusammengehörenden Klammern in der **gleichen Spalte** stehen:

```
print ( ;
    if ( ;
        eq ( ;
            1 , 1 ;
        ) ;
        "Willi", "Eva";
    );
)
```

Fehler bei der Markierung von Strings:

Auch dieser Fehler kommt recht häufig vor. Ein **vergessenes " -Zeichen** kann eine ganze **Befehlsfolge unbrauchbar** machen. Die durch diesen Fehler entstehenden **Folgefehler** führen leicht auf eine falsche Spur. Prüfen Sie auch hier Ihre Eingaben sorgfältig.

Fehler bei der Zahleneingabe in komplexen Funktionen:

Zahlen dürfen nur mit Dezimalpunkten eingegeben werden. Wenn Sie versehentlich ein Komma verwendet haben, werden aus **einer** Zahl **zwei** Parameter. Dies kann zu Parameterfehlern führen oder bei einigen Kommandos zu falschen Berechnungen.

Alle **anderen Fehler** werden Sie mit Hilfe der Fehlermeldungen **leicht korrigieren** können.

Bedienungsanleitung: SearARep

Sie finden trotz aller Bemühungen den Fehler nicht?

Versuchen Sie den Fehler zu **isolieren**, indem Sie eine **neue Konvertiertabelle** erzeugen, die nur die **fehlerhafte Zeile(n)** enthält.

Erzeugen Sie zusätzlich eine **Testdatei**, die einen **kurzen Testtext** für die Konvertierregel enthält.

Prüfen Sie, ob der Fehler immernoch auftritt. Wenn der Fehler nicht mehr auftritt, sollte die Ursache in den Testdaten liegen. Suchen Sie dann nach geeigneteren Testdaten.

Schicken Sie mir diese Daten: werner.perplies@weepee.de

5.1.4 Strategien zur Fehlervermeidung

Verwenden Sie einen **guten Editor**.

Kommentieren Sie Ihre Konvertiertabellen.

Verwenden Sie **verbundene Zeilen**.

Nutzen Sie **bestehende, ausgeteste** Teile **anderer Konvertiertabellen**.

Verwenden Sie **geschachtelte Konvertiertabellen**.

Fügen Sie **umfangreiche Ersetzungen** aus **externen Textdateien** ein.

Verwenden Sie **Macros**, die Sie in einer **separaten Konvertiertabelle definieren** und bei der **eigentlichen Verarbeitung ausführen**.

Bedienungsanleitung: SearARep

6 Beispieldateien

Im Verzeichnis ...**SearARep****Beispiele** finden Sie verschiedene Beispieldateien:

6.1 Muster-01.knv

In der Datei **Muster-01.knv** finden Sie eine Reihe von Konvertierregeln, die Ihnen als Vorlage für Ihre eigenen Konvertierregeln dienen sollen.

Alle Regeln sind ausführlich kommentiert. Zu dieser Tabelle gibt es keine Mustertexte.

6.2 Muster-02.knv/Muster-02.txt

Die **Muster-02-Konvertierungstabelle** enthält Konvertierregeln, die aus der beigefügten **Muster-02-Textdatei** (eine Corel-Ventura-Textdatei) mit einem Programmaufruf von **SearARep** eine **HTML-Datei** erzeugt, die Sie sich anschließend mit Ihrem **Webbrowser ansehen** können.

Starten Sie **SearARep**, wählen Sie die Konvertiertabelle **Muster-02.knv** aus und laden Sie anschließend die Datei **Muster-02.txt**.

Die Ergebnisdatei finden Sie unter ...**SearARep**_()**Beispiele**

Die in diesem Beispiel verwendeten Konvertierregeln sind wieder **umfassend kommentiert** und ihr **Schwierigkeitsgrad** reicht von **leicht** bis **erheblich**.

Besonderheiten sind:

sehr komplexe Funktion mit der **If-Funktion**

Verwendung von **Jokerfunktionen**

Ziffernjoker

Einfügung einer **Textdatei**

Umbenennung der **Ausgabedatei**

Die auf der HTML-Seite aufgerufenen Bilder wurden aus Platzgründen nicht dem Programm beigefügt.

Bedienungsanleitung: SearARep

**6.3 Muster-Arithmetik.knv/Muster-Arithmetik-2.knv/
Muster-Arithmetik.txt**

Diese Dateien zeigen Ihnen an einem konkreten Beispiel das Ausfüllen von einfachen Rechnungsformularen.

Starten Sie **SearARep**, wählen Sie die Konvertiertabelle **Muster-Arithmetik.knv** aus und laden Sie anschließend die Datei **Muster-Arithmetik.txt**.

Die Ergebnisdatei finden Sie unter ...**SearARep**_()**Beispiele**

Die in diesem Beispiel verwendeten Konvertierregeln sind wieder **umfassend kommentiert** und ihr **Schwierigkeitsgrad** reicht von **leicht** bis **erheblich**.

Besonderheiten sind:

Intensive Verwendung von **Variablen**

Anwendung von **Inlinekommandos**

Datumsfunktionen

Erzeugung von **Rechnungsnummern**

Bearbeiten von **Blöcken** mit der **Convert-Funktion**

Verwendung einer **zweiten Konvertiertabelle**

Arithmetische **Funktionen** (Addition, Multiplikation, Zwischensummen)

Ausfüllen von Feldern

Bedienungsanleitung: SearARep

7 SearARep in eine Vollversion umwandeln

Nach jedem Programmstart der Testversion können Sie **SearARep** in eine Vollversion umwandeln.

Nachdem Sie das Programm **SearARep** bestellt haben, erhalten Sie eine Mail, in der

der von Ihnen angegebene *Firmenname*

oder

der von Ihnen angegebene **Vor- und Nachname**, sowie ein **Registrierschlüssel** enthalten ist.

Verwenden Sie diese Daten unter

Beachtung der genauen Schreibweise

zur Umwandlung Ihres Programmes in eine Vollversion:

1. Schritt:

Registrieren wählen

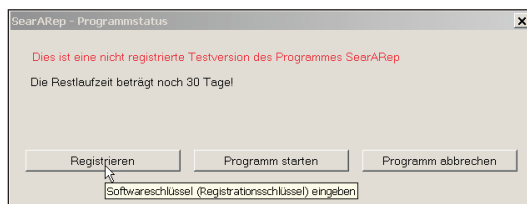


Abb. 43 Registrationsvorgang auswählen

Bedienungsanleitung: SearARep

2. Schritt:

Eingabe der Registrationsdaten:

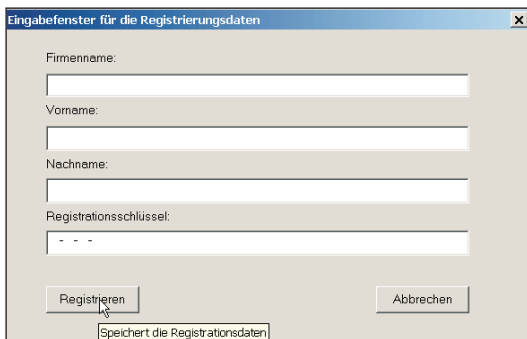


Abb. 44 Eingabe der Registrationsdaten

Als Firma:

Geben Sie bitte Ihre vollständige Firmenbezeichnung in **genau** der **Schreibweise** ein, die Sie bei der Bestellung verwendet haben.

Geben Sie den Registrierungsschlüssel **ohne** die Bindestriche ein.

Als Einzelperson:

Geben Sie bitte Ihren Vor- und Nachnamen in **genau** der **Schreibweise** ein, die Sie bei der Bestellung verwendet haben.

Geben Sie den Registrierungsschlüssel **ohne** die Bindestriche ein.

Bestätigen Sie die Eingabe mit

Registrieren

Bei Fehlern in der Eingabe erhalten Sie einen Fehlerhinweis und das Programm fordert Sie zur Wiederholung der Eingabe auf.

Wichtig:

Heben Sie diese Registrationsdaten gut auf, Sie benötigen sie bei einer Neuinstallation Ihres Rechners.

8 Anpassung von Textpad für SearARep

Zwölf Dateien mit einer Reihe von

Textbausteinen zur Erstellung einer Konvertiertabelle (.tcl)*

und **eine** Datei mit Befehlen

zur farbigen Markierung von Einträgen in Konvertiertabellen

für das Programm **Textpad** finden Sie unter

Programmordner\Textpad -> NN SearARep.tcl
Programmordner\Textpad -> SearARep.syn

Kopieren Sie diese Dateien in den Ordner von

Textpads Anwendungsdaten.

Üblicherweise heißt dieser Ordner unter

Windows 95/98/ME:

C:\Windows\Anwendungsdaten\Textpad

Windows NT/Windows 2000/Windows XP:

C:\Dokumente und Einstellungen\USER\Anwendungsdaten\TextPad

Zusätzlich finden Sie einen **Registrierschlüssel für Textpad** zur Erzeugung einer Dokumentenklasse für **SearARep-Konvertiertabellen** unter

Programmordner\Textpad -> Textpad-SearARep.reg

Achtung: Die Benutzung dieses Schlüssels geschieht auf eigene Gefahr. Sichern Sie unbedingt zuvor Ihre Registry.

Sie installieren diesen Schlüssel durch einen Doppelclick mit der Maus auf die Datei im Windows-Explorer.

Entnehmen Sie bitte **weitere Informationen** zur Arbeit mit **Textpad** der **Textpad-Dokumentation**.