

## UNIT REXXSTRING00.PP für Pascal

### Einleitung

Die Pascalunit REXXSTRING00.PP versucht die mächtigen REXX Zeichenkettenfunktionen auch für Pascalprogrammierer nutzbar zu machen.

Entwickelt und ausgetestet wurde REXXSTRING00.PP mit dem 32bit Freepascalcompiler Version 1.04 unter der Win32 Umgebung. Da aber nur Standardpascalbefehle wie „copy, pos, length...“ zum Einsatz kommen und bewusst auf den Typ „pchar“ verzichtet wurde, dürfte REXXSTRING00.PP universal, dh. ab TP4.0 funktionieren.

Fehler und Verbesserungsvorschläge richten Sie bitte an die Emailadresse [mueller@gm.fh-koeln.de](mailto:mueller@gm.fh-koeln.de) oder [muelut@web.de](mailto:muelut@web.de).

Lutz Müller, Januar 2002

### Nutzungsbedingungen

1. rexxstring00.pp ist Freeware und darf nur kostenlos weitergegeben werden
2. Die freie Verbreitung dieser Software ist erlaubt und sogar erwünscht
3. Für Funktionalität und Folgen der Nutzung von rexxstring00.pp wird keinerlei Haftung übernommen!
4. REXXSTRING00.PP darf nur mit diesem Text zusammen weitergegeben werden
5. Alle Rechte von REXXSTRING00.PP verbleiben beim Autor
6. Mit der Nutzung von REXXSTRING00.PP erkennen Sie die Punkte 1 bis 5 an

## Die Funktionen von REXXSTRING00.PP

REXX ist sehr stark im verarbeiten von Zeichenkette. Kein Wunder, den REXX "kennt" ja "nur" den Datentyp Zeichenkette. Mit den nachfolgenden Funktionen wurde der Versuch unternommen diese „Stärke“ von REXX Pacalbenutzern zur Verfügung zu stellen.

### function abbrev(s1, s2: string; laenge: integer): boolean

Testet, ob **s2** eine Abkürzung (Abbreviation) von **s1** ist. Es sollte auch noch eine **laenge** von Zeichen angegeben werden, bis zu der verglichen werden soll. Abbrev liefert „true“ bzw. „false“.

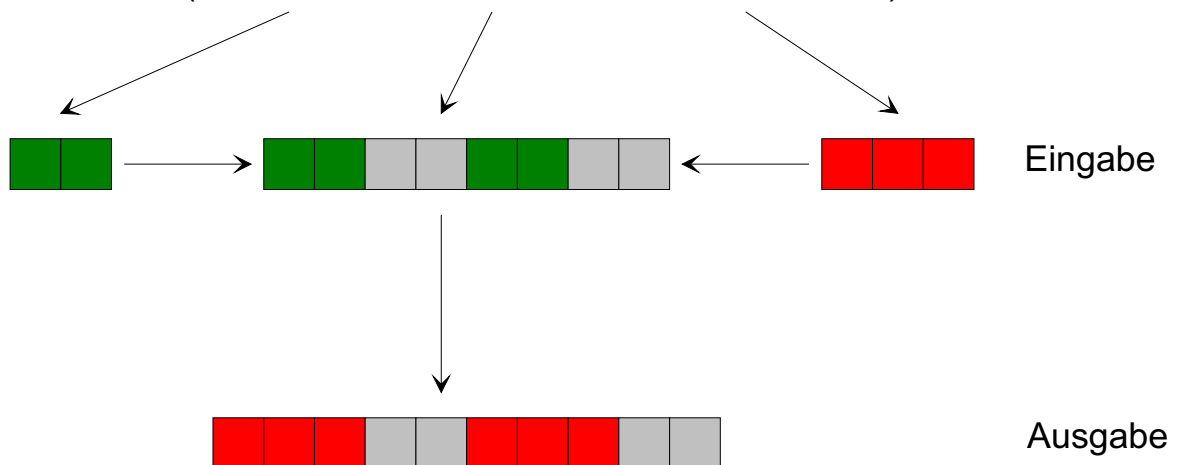
z.B.

```
ABBREV('FH Koeln', 'FH', 7)
```

ergibt „true“

### function changestr(s1, s2, s3: string): string

CHANGESTR(Zeichenkette1,Zeichenkette2,Zeichenkette3)



Tauscht in **Zeichenkette2**, falls enthalten, **Zeichenkette1** mit **Zeichenkette3**.

z.B.

```
CHANGESTR('FH', 'FH Koeln', 'Uni')
```

ergibt

```
Uni Koeln
```

### function centre(s1: string; laenge: integer; c1: char): string

Liefert eine Zeichenkette, die die in **laenge** angegebene Zahl von Zeichen hat. Grundlage für den Output ist **s1**. Es muss ein Füllzeichen angegeben werden.

z.B.

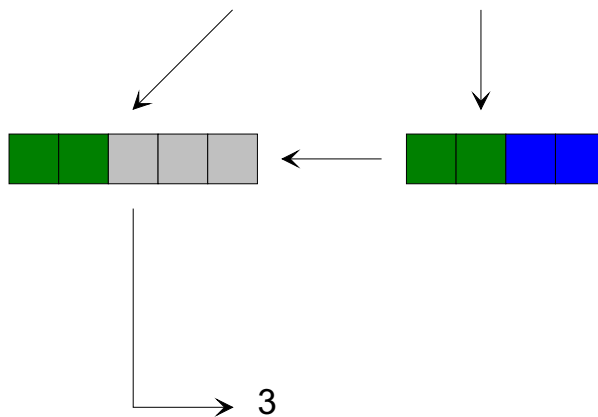
```
CENTRE('FH Koeln',20,'*')
```

ergibt

```
*****FH Koeln*****
```

### function compare(s1, s2: string; c1: char): integer

COMPARE(Zeichenkette1,Zeichenkette2[,Füllzeichen])



Eingabe

Ausgabe

Gibt die erste Stelle zurück, an der sich die beiden Zeichenketten unterscheiden. Optional kann angegeben werden, mit welchem **Füllzeichen** (c1) der kürzere der Zeichenketten aufgefüllt werden soll, wenn die zwei nicht gleich lang sind.

z.B.

```
COMPARE('Abt. Gummersbach','Abt. Koeln',' ')
```

ergibt 6

### function copies(s1: string; n: integer): string

Gibt **s1** so oft aus, wie Sie wollen. Nur kleiner 1 darf die Zahl nicht sein.

z.B.

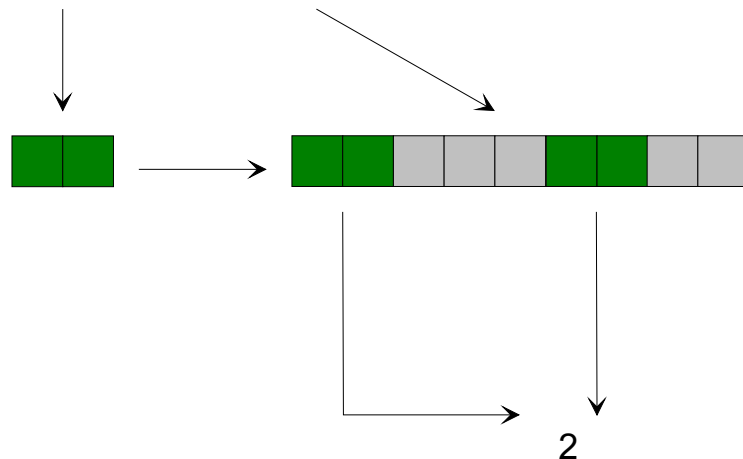
```
COPIES('FH Koeln ',3)
```

ergibt

```
FH Koeln FH Koeln FH Koeln
```

**function countstr(s1, s2: string): integer**

COUNTSTR(Zeichenkette1,Zeichenkette2)



Eingabe

Ausgabe

Zählt wie oft **Zeichenkette1** in **Zeichenkette2** enthalten ist.

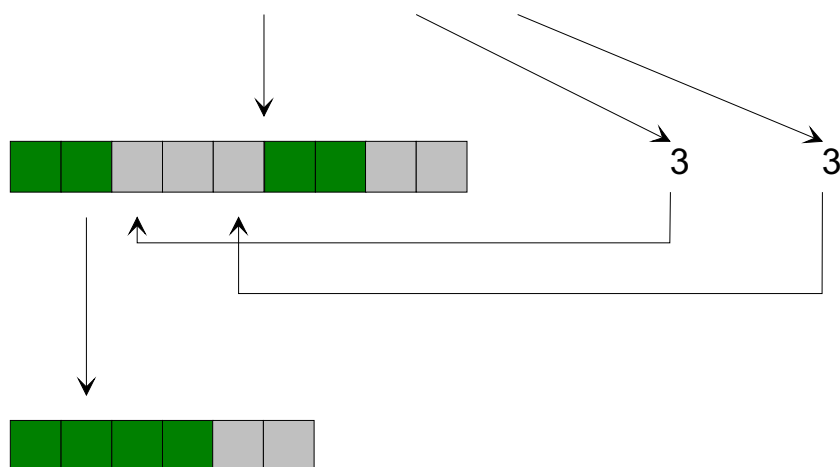
z.B.

COUNTSTR('KoeLn','FH KoeLn - Uni KoeLn')

ergibt 2

**function delstr(s1: string; posi, laenge: integer): string**

DELSTR(Zeichenkette,Pos[,Länge])



Eingabe

Ausgabe

Diese Funktion schneidet von **s1** alles ab dem Zeichen, dessen Position mit **Pos** angegeben ist, ab. **Laenge** gibt an wie viele Zeichen herausgeschnitten werden sollen.

z.B.

DELSTR('FH KoeLn - Uni KoeLn',4, 5)

ergibt

FH - Uni KoeLn

### function delword(s1: string; wposi, wanz: integer): string

Diese Funktion schneidet von **s1** alles ab dem Wort, dessen Position mit **wposi** angegeben ist, ab. Diese Teilzeichenkette ist **wanz**-Wörter lang.

z.B.

```
DELWORD('FH Koeln - Uni Koeln',3,3)
```

ergibt

```
FH Koeln
```

und z.B.

```
DELWORD('FH Koeln - Uni Koeln',2,1)
```

ergibt

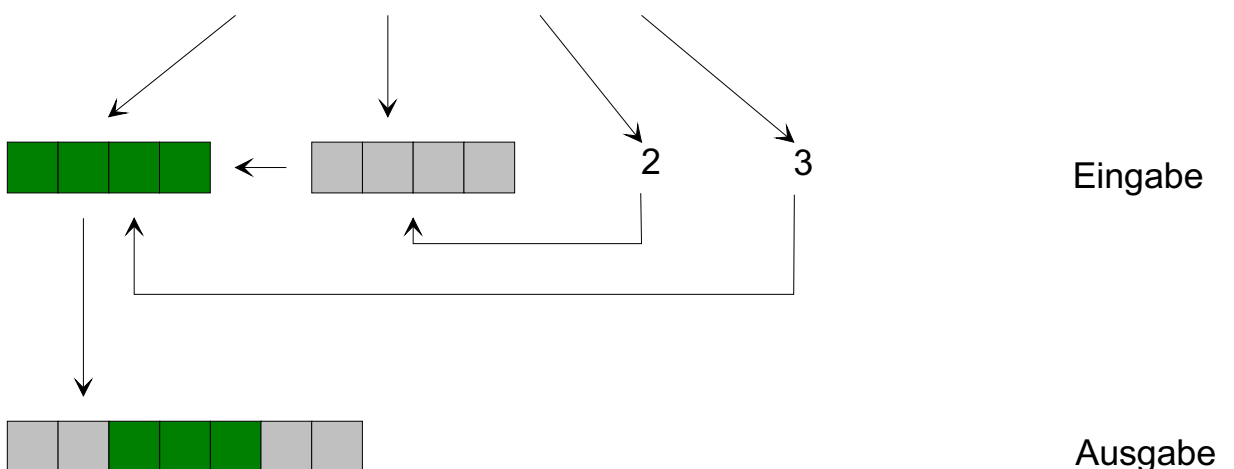
```
FH - Uni Koeln
```

### function firstpos(s1, s2: string; posi: integer): integer

Sucht nach dem ersten Auftreten von **s1** in **s2** und gibt die Position aus. 0, wenn nicht gefunden. Eine **Start**-Position des Suchvorgangs kann mit **posi** angegeben werden.

### function insert(s1, s2: string; posi, laenge: integer; c1: char): string

INSERT(Quellstring,Zielstring[,Start,Länge,Füllzeichen])



Fügt den **Quellstring** (s1) ab der **Start**-Position (posi) in den **Zielstring** (s2) ein. **laenge** gibt an wie viele Zeichen von **s1** eingefügt werden. Die Anzahl der Zeichen kann auch mit **Füllzeichen** erreicht werden.

z.B.

```
INSERT('QQQQ','ZZZZ',2,3,' ')
```

ergibt

```
ZZQQQZZ
```

### **function lastpos(s1, s2: string; posi: integer): integer**

gibt die Position des Zeichens an, an der **s1** zum letzten Mal in einer **s1** auftritt. **posi** gibt die Position an ab der **s1** durchsucht wird. Z.B.

```
LASTPOS( 'K', 'FH Koeln', 1)
```

ergibt 4

### **function left(s1: string; laenge: integer): string**

ergibt den linken Teil einer **Zeichenkette** (s1), bis zu dem Zeichen aus welches in **laenge** angegeben ist.

z.B.

```
LEFT( 'FH Koeln', 2)
```

ergibt

FH

### **function overlay(s1, s2: string; posi, laenge: integer; c1: char): string**

Damit wird eine Zeichenkette über eine andere gelegt. Im Gegensatz zu INSERT() geht dabei ein Teil des Zielstrings verloren. Das entspricht so ungefähr dem Überschreibmodus in einem Editor oder einer Textverarbeitung.

### **function reverse(s1: string): string**

Dreht **s1** um.

z.B.

```
REVERSE( ' nleoK HF')
```

ergibt

FH Koeln

### **function right(s1: string; laenge: integer): string**

Wie LEFT(), nur dass der rechte Teil abgeschnitten wird.

### **function space(s1: string; anz: integer; c1: char): string**

Damit können zwischen einzelnen Wörtern in **s1** beliebig viele Leerzeichen (oder andere Füllzeichen) eingefügt werden.

z.B.

```
SPACE( 'FH Koeln', 10, ' - ')
```

ergibt

FH-----Koeln

**function strip(s1: string; c1: char): string**

STRIP entfernt Zeichen **c1** aus **s1**.

z.B.

```
STRIP('---FH Koeln---','-')
```

ergibt

```
FH Koeln
```

**function substr(s1: string; posi, laenge: integer): string**

Damit kann eine Teilzeichenkette aus einer **Zeichenkette** (s1) übernommen werden, und zwar ab Position **posi**, mit angegebener **laenge**.

z.B.

```
SUBSTR('FH Koeln - Uni Koeln', 4, 5)
```

ergibt

```
Koeln
```

**function subword(s1: string; wposi, wanz: integer): string**

Ähnlich wie SUBSTR, nur entnimmt SUBWORD ganze Wörter aus Zeichenkette.

z.B.

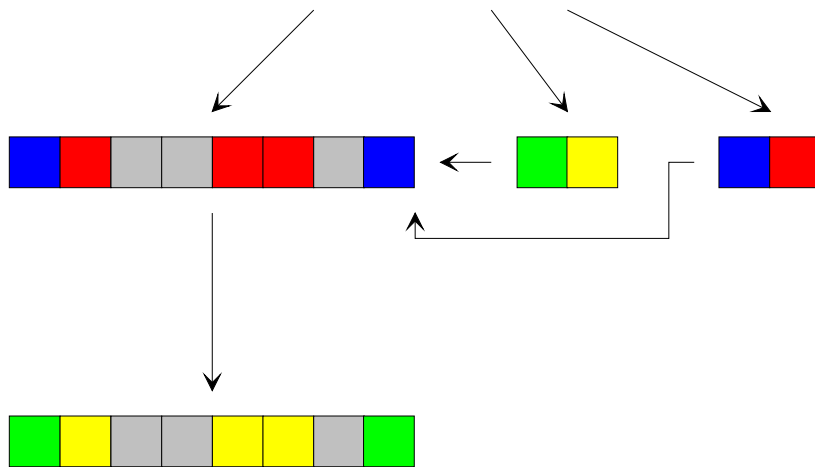
```
SUBWORD('FH Koeln - Uni Koeln', 1, 2)
```

ergibt

```
FH Koeln
```

## function translate(s1, rein, raus: string): string

TRANSLATE(Zeichenkette[,rein,raus,Füllzeichen])



Eingabe

Ausgabe

Leider kann dieser Befehl keine Texte in andere Sprachen übersetzen. Aber praktisch kann er trotzdem sein. Er vergleicht die in **raus** angegebenen Zeichen mit der Zeichenkette. Findet er eines, ersetzt er es durch das Zeichen, das in **rein** an gleicher Stelle angegeben ist.

z.B.

```
TRANSLATE('YX Koeln', 'HF', 'XY')
```

ergibt

```
FH Koeln
```

## function verify(s1, liste: string): integer

Überprüft, ob **s1** nur Zeichen aus der **Liste** enthält. Wenn ja, ist das Ergebnis 0, ansonsten die erste Position, an der ein fremdes Zeichen gefunden wurde.

z.B.

```
VERIFY('FH Koeln', 'HF nleKo')
```

ergibt 0

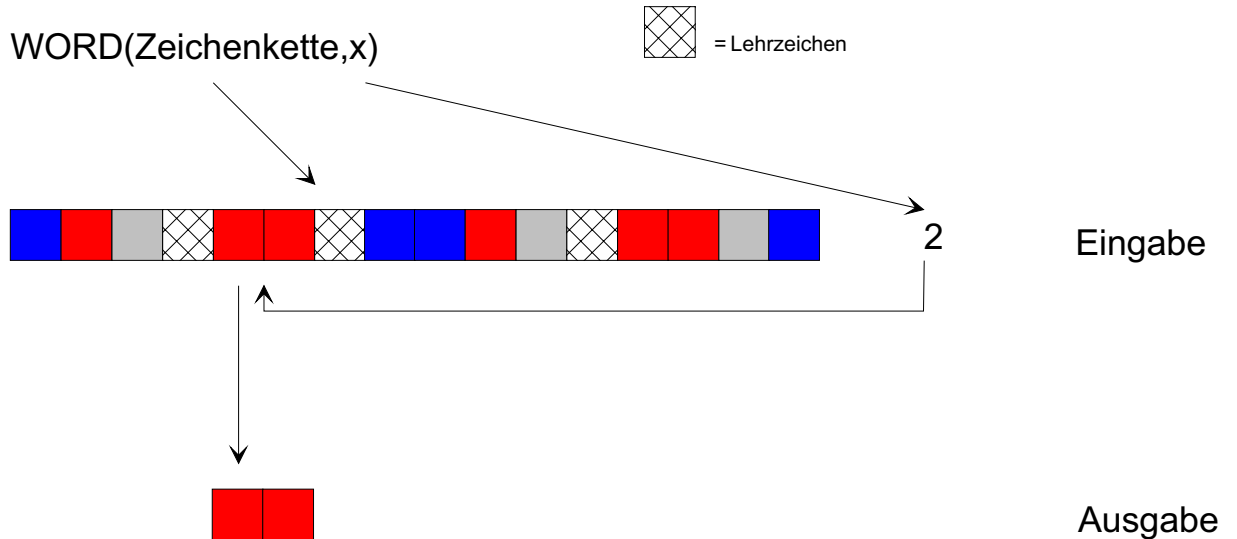
und z.B.

```
VERIFY('FH Koeln - Uni Koeln', 'HF nleKo')
```

ergibt 10



## function word(s1: string; wposi: integer): string



Gibt das **wposi**-te Wort aus der Zeichenkette aus, oder eine leere Zeichenkette, wenn so viele Wörter gar nicht vorhanden sind.

z.B.

```
WORD('FH Koeln - Uni Koeln',2)
```

ergibt

```
Koeln
```

## function wordindex(s1: string; wposi: integer): integer

Gibt die Position in Zeichen aus, an der das **wposi**-te Wort steht.

z.B.

```
WORDINDEX('FH Koeln Abt. Gummersbach', 4)
```

ergibt 15. (Das 4te Wort beginnt mit dem 15ten Zeichen)

## function wordlength(s1: string; wposi: integer): integer

Gibt die **Länge** des an Position **wposi** stehenden Wortes zurück oder 0, wenn es nicht so viele Wörter gibt.

z.B.

```
WORDLENGTH('FH Koeln Abt. Gummersbach', 4)
```

ergibt 11.

## function wordpos(s1, s2: string): integer

Liefert die Wortnummer des Wortes ab dem **s1** in **s2** beginnt.

z.B.

```
WORDPOS('Uni Koeln','Ich besuchte die Uni Koeln im Sommer.')
```

ergibt 4.

**function words(s1: string): integer**

Liefert die Anzahl der Wörter in Zeichenkette.

z.B.

```
WORDS( 'FH Koeln Abt. Gummersbach' )
```

ergibt 4.

**function xrange(c1, c2: char): string**

Liefert eine Zeichenkette, die alle ASCII-Codes von **c1** bis **c2** enthält.

z.B.

```
XRANGE( 'A', 'Z' )
```

ergibt

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```