Grundlagen der Informatik

Arbeit mit Zeichen und Zeichenketten

Prof. Dr.-Ing. Thomas Wiedemann
Fachgebiet Informatik / Mathematik

Überblick Textfunktionen

- Textbearbeitung mit VB
  - Motivation - Warum werden Textoperationen immer wichtiger ?
  - Typische Aufgaben bei der Stringbearbeitung

- Was ist Text in der EDV ? - Realisierung in VB
- Definition Textvariablen

- Zeichenorientierte Textfunktionen
- Mengenbasierte Textfunktionen
- Spezialfunktionen

Textbearbeitung mit C

- Motivation - Warum werden Textoperationen immer wichtiger ?
  - Informationsverarbeitung geht zunehmend auf Klartextdaten über.
  - Internet HTML ist eine reine Textgenerierung !!!!

- Typische Aufgaben bei der Stringbearbeitung
  - Numerische Daten in Texte einfügen
  - Textdaten analysieren und Teile extrahieren
  - Datenbankinformationen als HTML-Text kodieren
  - Befehlsanweisungen an Server zusammenbauen (SQL)

Prinzipielle Kodierung von Zeichen und Texten

- In den meisten Prozessoren gibt es bis auf Kopierbefehle keine speziellen Befehle zur Textverarbeitung

Fazit:

- Zeichen und Texte müssen auf Zahlen abgebildet werden !
- Alle Textoperationen, wie Vergleichen oder Suche in Texten müssen auf Operationen mit Zahlen zurückgeführt werden !
- Die prinzipielle Vorgehensweise ist in allen heutigen Programmiersprachen gleich, nur die Details der Kodierung und Verarbeitung sind unterschiedlich !
- Der Austausch von Texten zwischen verschiedenen Programmiersprachen und Betriebssystemen ist nicht trivial und erfordert zum Teil erhebliche Konvertierungen !

Kodierung von Zeichen und Texten II

- Zu kodieren sind :
  - 26 Groß- und 26 Kleinbuchstaben (+7 deutsche Umlaute "äöüß")
  - 10 Ziffern von 0 bis 9
  - ca. 30 Sonderzeichen .,!?%$§"%+-*=(){} ...
  - Steuerzeichen zur Kodierung spezieller Befehle (neue Zeile, neues Blatt Papier)

  = etwa 120 Symbole
- Es würden daher 7 Bit (=128 Zustände) ausreichen !
- Eine buchstabenweise Textverarbeitung erfordert einen getrennten Speicherzugriff auf jeden einzelnen Buchstaben.
- Bei der alten 8bit-Rechentechnik war damit das Byte die sinnvolle Größe zur Speicher eines Zeichens.
- 8 Bit erlauben die Kodierung von 256 Zuständen
- Mit den zusätzlichen 128 Zuständen lassen sich zusätzliche Zeichen (z.B. Rahmen für Fenster oder Kyrillisch) definieren.

8-bit-Kodierung von Zeichen

- Die Abbildung der Zeichen auf die Zahlendarstellung wurde standardisiert, z.B. durch American Standards Council (ASC)
- Allgemein verwendet : ASCII 1-Byte Darstellung in C
  (American Standard Code for Information Interchange)
- Relativ willkürliche Unterteilung in Blöcke zu je 32 Zeichen
  - 1. Block: Code 0-31 - nicht druckbare Steuerzeichen
  - 2. Block: Code 32-63 – Sonderzeichen und Ziffern
  - 3. Block Code 64 –95 - Großbuchstaben
  - 4. Block Code 96 –127 - Kleinbuchstaben
  - 5.-7. Block ab Code 128 spezielle Zeichen (nicht fest definiert !!!)

(Genaue Definition siehe Arbeitsblatt ASCII-Tabelle)

Probleme bei der 8-bit-Kodierung von Zeichen

Probleme mit Fremdsprachen und speziellen Zeichen

- Im Deutschen sind die Umlaute und das ß nicht im Standardbereich enthalten (Amerikaner kennen kein äöü !)
  - Abbildung auf obere Codeebene (>128)
- kyrillische und arabische Buchstaben sind nicht im ASCII-Standard definiert
  - Abbildung auf obere Codeebene (>128) -> nach KOI8-Standard
  - dabei Konflikt mit eventuell definierten Pseudografikzeichen oder Sonderzeichen anderer Sprachen
- Die mehreren Tausend asiatischen Schriftzeichen lassen sich mit dem ASCII-System nicht direkt darstellen
  - Umschreibung durch Kombination von ASCII-Sequenzen

Probleme mit deutschen Umlauten

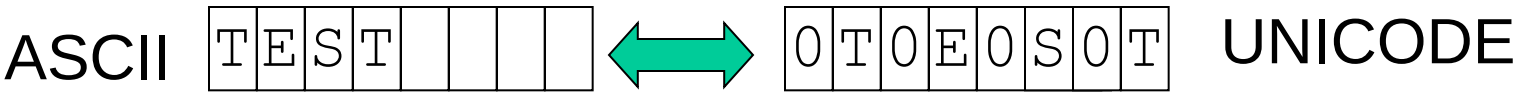
Leider ist auch die ASCII-Darstellung der deutschen Umlaute NICHT eindeutig.

- Unter DOS wurde mit den erweiterten Codepages der Seite 850 von IBM gearbeitet.
  - Der Buchstabe ä hat hier den Code 132
- Windows verwendet die ASCII-Kodierung nach dem ANSI-Standard:
  - Die ersten 128 Codes sind gleich !
  - Die Codes ab 128 weichen leider von der IBM-Definition ab !
  - Der Buchstabe ä hat hier den Code 228. Eine Anzeige dieses Buchstabens in einem DOS-Programm führt zu õ ! Analog ergeben die anderen Umlaute fehlerhafte Ausgaben (ü-> ³ ö -> ÷)
  - Auch bei Datenübernahmen oder Programmen kommt es zu dem Effekt !
- Lösung: Konvertierungsprogramme zum Tausch der Codes
- Achtung: Ähnliche Probleme entstehen bei der Kommunikation oder Arbeit mit UNIX-Programmen !

16-bit-Kodierung von Zeichen mit Unicode

Zukünftige Option zur Vermeidung der Probleme:

- Unicode 2-Byte Darstellung zur Erfassung von Sonderzeichen, virtuelle Tasten (z.B. Funktionstasten), Tastenkombinationen oder allgemein fremdsprachige Zeichen (diakritische Zeichen, kyrillisch, arabisch, asiatische Sprachen)
- Codetabelle wird in 256 Blätter mit jeweils 256 Zeichen unterteilt
- der ASCII-Code ist eine Untermenge auf dem 0. Blatt
  - ASCII-Texte lassen sich dadurch relativ einfach durch Einfügen einer zusätzlichen 0 nach Unicode wandeln (und umgekehrt)



- Unicode-Zeichensätze enthalten zusätzliche Angaben zur Kombination oder Zerlegung von Zeichen

Weitere Kodierungsarten von Zeichen

- EBCDIC 1-Byte Darstellung (Grossrechner)
- 7bit –Kodes (Lochstreifenformat) = 1. Hälfte ASCII
- 4bit – BCD-Darstellung von Zahlen
  - jeweils 2 Ziffern werden mit 2*4bit in einem Byte gespeichert
  - leichter konvertierbar, jedoch kompliziertere Rechnung
- 6bit – base64 - "Verschlüsselung" im Internet
  - Ziel: Übertragung der Information im Textformat durch Kodierung allein mit lesbaren Zeichen (32...A...Z)
  - aus 3 Zeichen a einem Byte werden 4 Zeichen mit jeweils 6 bit Information generiert (Aufteilen von 24 Bit auf 4 Byte)

Definition von Textvariablen in VB

Das Schlüsselwort für Texte in VB ist **String** (engl. Text)

Als Kurzform kann unmittelbar nach der Variable auch $ geschrieben werden !

(bei Integer-Variablen entsprechend %)

Definition von String in Normal- und Kurzform, auch als Feld :

```
Dim   s as String,  m$ ,    texte(100) as String
```

Zuweisung von Texten an Stringvariablen wie bei Zahlenwerten:

```
s = "Das ist ein "     m = "Fahrrad"  texte(2)="rotes"
s = "" :   Rem Das ist ein leerer Text !
```

Zeichenorientierte Funktionen

Der Zahlenwert des ersten Buchstabens eines Textes kann mit der Funktion

Asc(Text as String) as integer (von ASCII abgeleitet)

generiert werden.

Asc("1") liefert 49 = 31H (Hexdezimal)
Asc("2") liefert 50 = 32H (Hexdezimal)
Asc("A") liefert 65 = 41H (Hexdezimal)
Asc("B") liefert 66 = 42H (Hexdezimal)
Asc("a") liefert 97 = 61H (Hexdezimal)
Asc("b") liefert 98 = 62H (Hexdezimal)

Die umgekehrte Funktion ist **chr()** (von Character (engl. Zeichen) abgeleitet)

Diese erzeugt zu einem Zahlencode das entsprechende Zeichen !

chr(97) erzeugt ein einzelnes "a"

Achtung: Mit chr( ) können auch alle Sonderzeichen generiert werden (siehe Folgeseite) !

Sonderzeichen in VB generieren

Die ersten 32 Zeichen aus der ASCII-Tabelle sind mit Sonderfunktionen belegt. Das zugehörige Zeichen kann in Texten nicht direkt geschrieben , sondern muß mit der Funktion chr() eingefügt werden:

Die wichtigsten Code sind

chr(13) & chr(10) = entspricht Entertaste und erzeugt Zeilenumbruch

chr(9) = entspricht TAB-Taste

chr(7) = entspricht BELL = kurzer Signalton

Hinweis: Da das "-Zeichen bereits für die Begrenzung von Texten verwendet wird, muß ein entsprechendes Zeichen IM TEXT durch zwei "" realisiert werden. Damit wird ein in Anführungsstrichen eingeschlossener Text zu :

"Das ist ja zum ""Totlachen"" und ... "

Konvertierung von ganzen Texten von/nach Zahlenwerten

Sehr häufig müssen Texte mit numerischen Werten kombiniert werden oder in Zahlenwerte verwandelt werden :

Wandlung einer Zahl nach Text mit : **Str** (Abk. für String)

Str(23.5) erzeugt den Text " 23.5" (das erste Leerzeichen ist immer da!)

MERKE: In vielen Fällen erfolgt diese Wandlung auch automatisch !

d.h. bi „"Das Ergebnis ist „ & num erfolgt die Konvertierung direkt!

mit **Trim** können führende oder nachfolgende Leerzeichen entfernt werden

Trim (" 23.5 ") entfernt alle Leerzeichen und gibt "23.5" zurück

Wandlung eines Textes nach einer Zahl mit : **Val** (Abk. für Value)

i = val("123") : Rem i erhält den Zahlenwert 123

ACHTUNG: Val() versucht so viele Zeichen wie möglich in eine Zahl zu konvertieren und bricht bei Problemen ab ! Falls der Text keine Zahl darstellt, erzeugt Val den Wert 0 !!!

i = val("12m3") : Rem i erhält den Zahlenwert 12, da val bei m abbricht

i = val("Z123") : Rem i erhält den Zahlenwert 0, da val bei Z abbricht

Konvertierung von ganzen Texten von/nach Zahlenwerten

Formatierte Wandlung eines Textes

mit : **format** (Variable, Formatdefinition)

i = Format(d, "# ### ##0.00##") : erzwingt Ausgabe mit Leerstellen und 2 Nachkommastellen : 2 222 222,00

Formatkodierung: # stellt optionale Zahlenposition bereit
0 erzwingt Ausgabe einer Ziffer (ggf. 0)

-> zahlreiche, weitere Optionen für **Datumsformatierung**, z.B.

d = now() : rem Hole aktuelles Datum und Uhrzeit

i = Format(d, “hh:mm T.M.J") erzeugt Uhrzeit / Datum, z.B. “09:31 11.12.18“

(siehe Online-Hilfe für komplette Liste)

Zerlegen / Extrahieren von Textbestandteilen

Oft müssen zusammengesetzte Texte wieder in einzelne Stücke zerlegt werden.

Dazu stehen folgende Funktionen bereit :

s = **Left**(“ABCDE“,3) - extrahiert die 3 linken Zeichen (“ABC“)

s = **Right**(“ABCDE“,2) - extrahiert die 2 rechten Zeichen (“DE“)

s = **Mid**(“ABCDE“,3,2) - extrahiert ab 3. Position 2 Zeichen (“CD“)

s = **Mid**(“ABCDE“,3)- extrahiert ab 3. Position bis ENDE !! (“CDE“)

Zum Prüfen, ob in einem Text1 ein Text2 enthalten ist, dient die Funktion instr(text1,text2) , welche die Position des 2. Textes zurückgibt oder eine 0 bei Nichtvorhandensein liefert!

Bsp.: instr(“Das ist ein Fahrrad“ , “ist“) liefert 5 als Ergebnis

instr(“Das ist ein Fahrrad“ , “Auto“) liefert 0 als Ergebnis

Mit den diesen Funktionen können fast alle Transformationen von Texten durchgeführt werden (Mitschrift VL) !