9.10 Konvertierungsfunktionen

In vielen Programmen werden Konvertierungen zwischen verschiedenen Daten-Typen notwendig, wenn zum Beispiel Werte aus einer TextBox als Zahlen oder als Datum typ-gerecht weiter verarbeitet werden sollen. Beachten Sie bei allen Konvertierungsfunktionen den Daten-Typ der Argumente und den Daten-Typ des Funktionswertes sowie die Hinweise zur Lokalisierung.

9.10.1 Tabelle ausgewählter Konvertierungsfunktionen

Funktion	Beschreibung
CBool (Expr) AS Boolean	Konvertiert einen Ausdruck in einen Wahrheitswert (True oder False). Ein Ausdruck ist garantiert falsch, wenn ein Wahrheitswert Falsch ist oder eine Zahl 0 ist oder die String-Länge 0 ist oder ein NULL-Datum vorliegt oder ein Null-Objekt.
CByte (Expr AS Variant) AS Byte	Konvertiert einen Ausdruck in ein Byte. Der Ausdruck wird zuerst in einen Integer-Wert konvertiert. Überschreitet der Integer-Wert den Byte-Bereich [0..255] , dann wird er (modulo 256) in diesen Bereich transformiert.
CDate (Expr AS Variant) AS Date	Konvertiert einen Ausdruck (Zahl vom Daten-Typ Integer oder Float) in einen Datum/Zeit-String. CDate nutzt die aktuelle Lokalisierung nicht!
CFloat (Expr AS Variant) AS Float CFlt (Expr AS Variant) AS Float	Konvertiert einen Ausdruck in eine Fließkommazahl. Vorsicht! Diese Funktion nutzt die eingestellte Lokalisierung nicht. Es muss als Dezimal-Separator ein Punkt gesetzt werden!
CInt (Expr AS Variant) AS Integer CInteger (Expr AS Variant) AS Integer	Konvertiert einen Ausdruck in eine Zahl vom Daten-Typ Integer.
CLong (Expr AS Variant) AS Long	Konvertiert einen Ausdruck in eine Zahl vom Daten-Typ Long. Eine Bereichsüberschreitung wird nicht erkannt.
CShort (Expr AS Variant) AS Short	Konvertiert einen Ausdruck in eine Zahl vom Daten-Typ Short. Der Ausdruck wird zuerst in einen Integer-Wert konvertiert. Überschreitet der Integer-Wert den Short-Bereich [-32.768 .. +32.767], dann wird er in diesen Bereich transformiert. Dabei werden die geringst-signifikanten 16 Bits aus *Expr* extrahiert und im Sinne des Zweierkomplements als Ganzzahl interpretiert.
CSingle (Expr AS Variant) AS Single	Konvertiert einen Ausdruck in eine Zahl vom Daten-Typ Single. Diese Funktion nutzt die eingestellte Lokalisierung nicht!
CStr (Expr AS Variant) AS String CString (Expr AS Variant) AS String	Konvertiert einen Ausdruck in einen String. Diese Funktion nutzt die eingestellte Lokalisierung nicht!
CVar (Expr) AS Variant CVariant (Expr) AS Variant	Konvertiert einen Ausdruck in einen Variant. Dies ist nützlich, wenn das Ergebnis einer Funktion vom Datentyp der Argumente abhängig ist.
Str$ (Expr) AS String Str (Expr) AS String	Konvertiert eine Zahl *oder* ein Datum in einen String. Diese Funktion beachtet die eingestellte Lokalisierung!
DConv$ (String AS String) AS String DConv (String AS String) AS String	Konvertiert einen String vom System-Zeichensatz nach UTF-8, den Desktop-Zeichensatz.
SConv$ (String AS String) SConv (String AS String)	Konvertiert einen String vom Desktop-Zeichensatz (der UTF-8 sein sollte) in den System-Zeichensatz.
Val (String)	Konvertiert einen String in eine Zahl *oder* in ein Datum. Diese Funktion berücksichtigt die eingestellte Lokalisierung!

Tabelle 9.10.1.1: Übersicht zu den Konvertierungsfunktionen

Der Umwandlungsalgorithmus für die Val()-Funktion ist der folgende:

- Wenn der String als Datum und Uhrzeit (mit Datums- oder Zeit-Trennzeichen entsprechend der Lokalisierung) interpretiert werden kann, wird das Datum oder die Zeit zurückgegeben.
- Wenn der String als Fließkommazahl interpretiert werden, dann wird die Zahl zurückgegeben.
- Wenn der String als 64-Bit-Zahl interpretiert werden kann, dann wird diese Zahl vom Daten-Typ Long zurückgegeben. Ansonsten wird – wenn die Zeichenfolge als ganze Zahl interpretiert werden kann – diese ganze Zahl zurückgegeben.

- Wenn String = "True" oder String = "False" ist, so werden die passenden Wahrheitswerte "True" oder "False" zurückgegeben.
- In allen anderen Fällen wird NULL zurückgegeben.

9.10.2 Konvertierungsfunktionen Zeichensätze

```
UmgewandelterString = Conv$ ( String AS String , QuellZeichensatz AS String , ZielZeichensatz AS String )
UmgewandelterString = Conv  ( String AS String , QuellZeichensatz AS String , ZielZeichensatz AS String )
```

- Wandelt einen String von einem Zeichensatz in einen anderen um.
- Ein Zeichensatz wird durch einen String wie "ASCII", "ISO-8859-1", oder "UTF-8" repräsentiert.

- Der Gambas-Interpreter benutzt intern den UTF-8-Zeichensatz.
- Der vom System benutzte Zeichensatz wird in der Eigenschaft 'System.Charset' zurückgegeben.
- Zukünftig werden vermutlich alle Linux-Systeme auf UTF-8 basieren.
- Den Zeichensatz, der vom grafischen Benutzerinterface verwendet wird, liefert Desktop.Charset. Dieser sollte UTF-8 sein.
- Die Umwandlung beruht intern auf der iconv()-GNU-Library-Funktion.

9.10.3 Beispiele

```
Dim sNumber As Single

Print CBool(0), CBool(1), CBool("Gambas"), CBool(""), CBool(Null), CBool(Date(2000, 2, 29))

Print CByte("23"), CByte("257"), CByte(True)

Print "CDate(2488913) = "; CDate(2488913)
Print "UnixTimeStamp  = "; DateDiff(CDate("1/1/1970"), Now, gb.Second)
Try CDate("12.2. 2014 12:45:00")
If Error.Code = 6 Then Print "Der Ausdruck '"; "12.2. 2014 12:45:00"; "' kann nicht konvertiert werden!"
Print "Datum: ", Format(CDate(Val("1.9.2012")), "d. mmmm yyyy")
Print "Tage seit 1.1. 1970 = "; DateDiff(CDate("1/1/1970"), Now, gb.Day) ' 16224

Print CFloat("+3.1416"), CFloat(Now())

Print CInt("17"), CInt(True), CInt(Pi(1 / 7)), CInt(3.8), CInt(-7.998), CInt(-7.1), CInt(Now)

Print "CLong(5 ^ (2 ^ 2)) = "; CLong(5 ^ (2 ^ 2))
Print "CLong(2^62) = "; CLong(2 ^ 62)

Print CShort(20 < 6), CShort(False), CShort(True), CShort(DateDiff(CDate("1/1/1970"), Now, gb.Day))

Try sNumber = CSingle(DateDiff(CDate("1/1/1970"), Now(), gb.Millisecond))
If Error Then Print "ÜBERLAUF"
Print CSingle(Pi), CFloat(Pi), CSingle(Exp(11)), CSingle("+355.11")

Print CStr(-99), CStr(Pi(Pi())), CStr(355 / 113), CStr(Now)

Print "Object.Type(CVariant([2, 3, 5])) = "; Object.Type(CVariant([2, 3, 5]))
Print "Object.Type(CVariant([\"2\",\"3\",\"5\"])) = "; Object.Type(CVariant(["2", "3", "5"]))
Print "Object.Type([CVariant(\"a\"),\"b\",\"c\"]) = "; Object.Type([CVariant("a"), "b", "c"])

Print "Val(\"True\") = "; Val("True"), "Val(\"False\") = "; Val("False")
Print Format(Val("12.11.2013"), "d. mmmm yyyy")
Print "Val(\"123,456\") = "; Val("123,456"); " (Zahl mit Komma als Dezimal-Trennzeichen für DE.de)"
Print "Val(\"123.456\") = "; Val("123.456"); " (Punkt wird als Tausender-Trennzeichen interpretiert!)"
If Val("123.66") = Null Then Print "Val(\"123.66\") = NULL"; " (String nicht als Zahl interpretierbar.)"

Print Conv("Ärger in der Ödipus-Straße", System.Charset, "ISO 8859-15")
Print "Desktop-Zeichensatz = "; Desktop.Charset
Print "System-Zeichensatz  = "; System.Charset

False   True    True    False   False   True
23      1       255
CDate(2488913) = 30.05.2014 23:00:00
UnixTimeStamp  = 1401992840
Der Ausdruck '12.2. 2014 12:45:00' kann nicht konvertiert werden!
Datum:  1. September 2012
Tage seit 1.1. 1970 = 16226
3,1416  2488918,81065353
17      -1      0       3       -7      -7      2488918
CLong(5 ^ (2 ^ 2)) = 625
CLong(2^62) = 4611686018427387904
0       0       -1      16226
ÜBERLAUF
```

```
3,1415927       3,14159265358979        59874,140625    355,1099854
-99     9.86960440108936        3.14159292035398        06/05/2014 18:27:20.466
Object.Type(CVariant([2, 3, 5]))     = Integer[]
Object.Type(CVariant(["2","3","5"])) = String[]
Object.Type([CVariant("a"),"b","c"]) = Variant[]
Val("True") = True      Val("False") = False
12. November 2013
Val("123,456") = 123,456 (Zahl mit Komma als Dezimal-Trennzeichen für DE.de)
Val("123.456") = 123456 (Punkt wird als Tausender-Trennzeichen interpretiert!)
Val("123.66") = NULL (String nicht als Zahl interpretierbar.)
Ärger in der Ödipus-Straße
Desktop-Zeichensatz = UTF-8
System-Zeichensatz  = UTF-8
```