

Anweisungen zur Textbearbeitung im Modul Display

Funktionen	Beschreibung, Beispiel
S := Display.WhereX() S: INTEGER Z := Display.WhereY() Z: INTEGER	Liefert die aktuelle Position bzgl. der x-Achse (Spalte) zurück. Liefert die aktuelle Position bzgl. der y-Achse (Zeile) zurück.
B := Display.KeyPressed() B: BOOLEAN	Der Wahrheitswert B (TRUE / FALSE) gibt an, ob eine Taste auf der Tastatur gedrückt wurde oder nicht. REPEAT UNTIL Display.Keypressed();
C := Display.ReadKey() C: CHAR	Diese Funktion liefert als Ergebnis das entsprechende ASCII-Zeichen der zuletzt benutzten Taste. Ist es kein ASCII-Zeichen (Funktionstasten), so wird der Wert auf Null gesetzt. Ein nochmaliger Aufruf der Funktion legt den zugehörigen Tastaturcode in C ab.
B := Display.IsCursorOn() B: BOOLEAN	B wird auf TRUE gesetzt, wenn der Cursor eingeschaltet ist.
B := Display.IsColorSupported() B: BOOLEAN	Sollte das System keine Farbausgaben unterstützen, wird B auf FALSE gesetzt, d.h. alle Farbwerte sind automatisch 0.

Prozedur	Beschreibung, Beispiel
Display.SetWindowTitle(titel) titel: ARRAY ... OF CHAR	Setzt im Display-Fenster den Titel auf den angegebenen Text. Display.SetWindowTitle('Mein Fenster');
Display.GotoXY(s,z) s,z: INTEGER	Setzt den Cursor an die Position mit Spalte s und Zeile z. Display.GotoXY(10,3);
Display.WriteChar(c) c: CHAR Display.WriteStr(text) text: ARRAY ... OF CHAR Display.WriteSpaces(anz) anz: INTEGER Display.WriteInt(i,l) i: LONGINT l: INTEGER Display.WriteReal(r,l) r: REAL l: INTEGER Display.WriteLn()	Ausgabeeinstruktionen für die angegebenen Datentypen ab der aktuellen Cursorposition. Für die Zahltypen bestimmt der Parameter l die Länge der Ausgabe. Display.WriteSpaces(anz) gibt anz Leerzeichen aus. Setzt den Cursor in die neue Zeile.
Display.WriteCharXY(s, z, c) c: CHAR Display.WriteStrXY (s, z, text) text: ARRAY ... OF CHAR Display.WriteSpacesXY (s, z, anz) anz: INTEGER Display.WriteIntXY (s, z, i, l) i: LONGINT l: INTEGER Display.WriteRealXY (s, z, r, l) r: REAL l: INTEGER s, z: INTEGER	Analog zu den oben genannten Anweisungen erfolgt die Ausgabe an der Position mit Spalte s und Zeile z.
Display.CursorOn() Display.CursorOff()	Schaltet den Cursor ein bzw. aus.
Display.TerminalBell()	Ein kurzer Ton wird ausgegeben. (Wenn es funktioniert!)
Display.EditStr(text,l,code) text: ARRAY ... OF CHAR l: INTEGER code: CHAR	Die angegebene Zeichenkette text wird ausgegeben und kann im Anschluss verändert werden. Der Wert l bestimmt die Anzahl der Zeichen, die max. hinzugefügt werden können.

Display.ReadChar(c) c: CHAR Display.ReadStr(text) text: ARRAY ... OF CHAR Display.ReadInt(i) i: INTEGER Display.ReadLongInt(li) li: LONGINT Display.ReadReal(r) r: REAL Display.ReadLongReal(lr) lr: LONGREAL	Eingabeanweisungen für die angegebenen Datentypen ab der aktuellen Cursorposition.
Display.FlushKeyBuffer()	Tastaturpuffer wird gelöscht. Dies stellt sicher, dass der nächste Aufruf von Display.ReadKey() keine vorher benutzten Tasten auswertet.
Display.ClrScr()	Löscht den Bildschirm in der aktuellen Hintergrundfarbe (Standard: weiß).
Display.SetForeColor(rot,grün,blau) Display.SetBackColor(rot,grün,blau) rot,grün,blau: INTEGER	Setzt die Vorder- bzw. Hintergrundfarbe durch Mischen der Farbanteile. Der Wert für rot,grün,blau liegt jeweils zwischen 0 und 255.
Display.GetForeColor(rot,grün,blau) Display.GetBackColor(rot,grün,blau) rot,grün,blau: INTEGER	Ermittelt die aktuelle Vorder- bzw. Hintergrundfarbe.

```

MODULE TESTDISP;
  IMPORT D:=Display;
  VAR i: INTEGER; j,k: LONGINT;
      x: REAL;
      c: CHAR;
  PROCEDURE ProgMain*;
  BEGIN
    D.SetWindowTitle("Test Display");
    D.SetBackColor(20,200,200);D.ReadInt(i,10,c);
    REPEAT
      D.ClrScr();
      D.SetForeColor(200,100,100);
      D.WriteStrXY(10, 10,"AUSGABE EINES TEXTES");
      FOR j:=1 TO 100000 DO k:=j+1 END;
      D.ClrScr();
      D.SetForeColor(200,200,100);
      D.WriteStrXY(20, 20,"AUSGABE EINES TEXTES")
    UNTIL D.KeyPressed();
    REPEAT UNTIL D.KeyPressed();
  END ProgMain;
END TESTDISP.

```