22 Programmierung der Tastatur

Gliederung

22.1 Die erweiterten Tastaturcodes

Mit der Standardfunktion *ReadKey* können alle 256 Zeichen des (von IBM erweiterten) ASCII-Zeichensatzes eingelesen werden; *ReadKey* liefert in diesem Falle genau ein Zeichen. Das trifft nicht zu für die Sondertasten, wie z.B. die Funktionstasten oder die Cursortasten und auch für Kombinationen von "normalen" Tasten mit den sogenannten Umschalttasten *Umsch = Shift*, *Alt* und *Strg = Ctrl*.

Die genannten Sondertasten oder die Tastenkombinationen mit ihnen liefern zwei Zeichen, wobei das erste Zeichen nach der MS-DOS-Vereinbarung das Null-Byte (Zeichen mit dem ASCII-Code 0, Chr(0), #0) ist und erst das zweite Zeichen, das in Turbo-Pascal unbedingt mit einem weiteren ReadKey-Aufruf eingelesen werden muß, die eigentliche Information über die Sondertaste oder die Tastenkombination enthält. Die Zuordnung des zweiten Zeichens zu der Sondertaste oder der Tastenkombination ist von MS-DOS festgelegt.

Die folgende Tabelle enthält den dezimalen Code des *zweiten* Zeichens des erweiterten Tastaturcodes und – falls im Textprogramm und auf dem Drucker darstellbar – das Zeichen selbst.

Taste	ohne	Umsch	Alt	Strg
A			030	s
B			048	0
C			046	.
D			032	
E			018	×
F			033	!
G			034	"
H			035	#
I			023	
J			036	$\overline{\$}$
K			037	%
L			038	&
M			050	2
N			049	1
O			024	$\underline{-}$
P			025	$\overline{\_}$
Q			016	
R			019	
S			031	
T			020	¶
U			022	
V			047	/
W			017	
X			045	-
Y			021	$\underline{\$}$
Z			044	,

Taste	ohne	Umsch	Alt	Strg
1			120 x	
2			121 y	
3			122 z	
4			123 {	
5			124 |	
6			125 }	
7			126 ~	
8			127	
9			128 Ç	
0			129 ü	
F1	059 ;	084 T	104 h	094 ^
F2	060 <	085 U	105 i	095 _
F3	061 =	086 V	106 j	096 `
F4	062 >	087 W	107 k	097 a
F5	063 ?	088 X	108 l	098 b
F6	064 @	089 Y	109 m	099 c
F7	065 A	090 Z	110 n	100 d
F8	066 B	091 [	111 o	101 e
F9	067 C	092 \	112 p	102 f
F10	068 D	093]	113 q	103 g
F11	133 à	135 ç	139 ï	137 ë
F12	134 å	136 ê	140 î	138 è
Cu-Li	075 K			115 s
Cu-Re	077 M			116 t
Cu-Ob	072 H			
Cu-Un	080 P			
Home	071 G			119 w
End	079 O			117 u
PgUp	073 I			132 ä
PgDn	081 Q			118 v
Ins	082 R			
Del	083 S			
Tab		015		
-			130 é	
PrtSc				114 r

Die **Umschalttasten** selbst (*Umsch*, *CapsLock*, *Alt*, *Strg* und *Scroll*) können nicht in dieser Weise abgefragt werden. Lösung dazu siehe Kapitel "Systemnahe Programmierung".

22.2 Demo-Programm 1

```
program Pas22021;  { Tastaturabfrage bei Sondertasten }
                   { Einleitendes Beispiel            }
uses
  CRT;

var
  Zeichen: Char;

begin
  TextBackground(Blue); TextColor(Yellow); ClrScr;

  Writeln('Zeicheneingabe. Ende mit Taste "Esc": ', #13#10);

  repeat
    TextColor(White);
    Zeichen := ReadKey; { Das erste Byte ist bei Sondertasten ... }

    if Zeichen = #0     { ... immer das Null-Byte »#0«           }
       then begin
              Write('Ein Sonderzeichen:  ');
              ;
              Zeichen := ReadKey;  { Das 2. Byte lesen,   }
              ;                    { wenn zuvor Null-Byte }
              case Zeichen of
              'K': Write('Cursor links.     2. Byte: ');
              'M': Write('Cursor rechts.    2. Byte: ');
              'H': Write('Cursor oben.      2. Byte: ');
              'P': Write('Cursor unten.     2. Byte: ');
              ';': Write('F-Taste F1.       2. Byte: ');
              '<': Write('F-Taste F2.       2. Byte: ');
              '=': Write('F-Taste F3.       2. Byte: ');
              else Write('Nicht abgefragt:           ');
              end;
            end
       else Write('"Normales" Zeichen: ');

    TextColor(Yellow);
    WriteLn(Zeichen);
  until Zeichen = #27;   { Zeichen "#27": Escape }
end.
```

22.3 Demo-Programm 2

```
program Pas22031;  { Programmierung der Tastatur, Scan-Codes         }

{ Dieses Demo-Programm verwendet die Cursortasten sowie die
  Funktionstasten F1, F2, F3 und F10. Diese Tasten zählen zu den
  Sondertasten und liefern nach Vorgabe von MS-DOS einen Doppelcode,
  wobei das erste Byte das Null-Byte "#0" ist.

  Mit dem Programm kann der Cursor innerhalb eines Bildschirmfensters
  positioniert werden. Mit der Funktionstaste "F2" kann an der Cursor-
  stelle ein Punkt gesetzt werden, wogegen mit "F3" durch Über-
```

```
  schreiben mit einem Blank ein eventuell gesetzter Punkt gelöscht
  wird. Die Funktionstaste "F1" ist für den Aufruf eines Hilfe-Textes
  vorgesehen. Mit der Funktionstaste "F10" soll die eigene künst-
  lerische Aktivität beendet und in eine automatische übergeführt
  werden.

  Bei allen folgenden Sondertasten ist nur das zweite Byte des
  Doppelcodes angegeben, da das erste Byte bei Sondertasten immer
  das Null-Byte "#0" ist. Das zweite Byte ist bei den meisten Sonder-
  tasten mit dem Scan-Code der Taste identisch.
}
uses
  CRT;

const
  F1  = #59;  { Taste F1:  Scan-Code dez 059, hex $3B, Zeichen ";"  }
  F2  = #60;  { Taste F2:  Scan-Code dez 060, hex $3C, Zeichen "<"  }
  F3  = #61;  { Taste F3:  Scan-Code dez 061, hex $3D, Zeichen "="  }
  F10 = #68;  { Taste F10: Scan-Code dez 068, hex $44, Zeichen "D"  }
  { Bei den folgenden 4 Tasten sind die Codes des zweiten Bytes }
  { nicht mit den Scan-Codes der Tasten identisch:              }
  CursorLinks   = #75;   { dez 075, hex $4B, Zeichen "K" }
  CursorRechts  = #77;   { dez 077, hex $4D, Zeichen "M" }
  CursorOben    = #72;   { dez 072, hex $48, Zeichen "H" }
  CursorUnten   = #80;   { dez 080, hex $50, Zeichen "P" }

  SpalteMin     = 20;
  SpalteMax     = 52;
  ZeileMin      = 12;
  ZeileMax      = 22;

  PunktSetzen   = #254;  { ASCII 254 oder irgendein anderes Zeichen }
  PunktLoeschen = #32;   { Leerzeichen, Blank                       }

var
  ZulaessigeZeichen: string[8];
  Zeile,
  Spalte:            Byte;
  ErstesByte,
  ZweitesByte:       Char;

procedure WriteXY(Spalte, Zeile: Byte; Meldung: string);
begin
  GotoXY(Spalte, Zeile);
  Write(Meldung);
end;

procedure Legende; { ---------------------------------------------- }
const
  L1 = 'Dieses Programm zeigt die Abfrage einiger Sondertasten.';
  L2 = 'Sondertasten sind unter anderen die Cursortasten und die';
  L3 = 'Funktionstasten. Weitere Sondertasten siehe Skriptum.';
  L4 = 'Im Gegensatz zu den anderen Tasten liefern die Sonder-';
  L5 = 'tasten einen Doppelcode, wobei das erste Byte das';
  L6 = 'Nullbyte "#0" ist.';
  L7 = 'Die Funktiontasten können mehrfach belegt werden, z.B. mit';
  L8 = '"Umsch", "Alt" und "Strg". Details siehe Skriptum.';
  Spalte = 10; { lokale Konstante !! }

begin
  WriteXY(Spalte,  3, L1);
```

```
  WriteXY(Spalte,  4, L2);
  WriteXY(Spalte,  5, L3);
  WriteXY(Spalte,  6, L4);
  WriteXY(Spalte,  7, L5);
  WriteXY(Spalte,  8, L6);
  WriteXY(Spalte,  9, L7);
  WriteXY(Spalte, 10, L8);
end;     { von Prozedur "Legende" }


procedure Bildschirmaufbau; { ------------------------------------ }
var
  AnzahlSpalten:          Byte;
  Doppelstrich Waagrecht,
  Blank String:           string;

begin
  TextBackground(Blue); TextColor(Yellow); ClrScr;

  AnzahlSpalten := SpalteMax - SpalteMin + 1;

  { Das Füllen eines Strings mit gleichen Zeichen ist in Pascal }
  { ziemlich umständlich, wie nachfolgend zu sehen ist ...      }
  FillChar(Doppelstrich Waagrecht, AnzahlSpalten + 1, '=');
      { String füllen und Längenbyte belegen }
  Doppelstrich Waagrecht[0] := Chr(AnzahlSpalten);

  FillChar(Blank String, AnzahlSpalten + 1, ' ');
      { String füllen und Längenbyte belegen }
  Blank String[0] := Chr(AnzahlSpalten);

  WriteXY(SpalteMin, 1, 'Demonstration Tastatur, Scan-Codes');
  TextColor(White);

  Legende;   { Aufruf der Prozedur "Legende" }

  WriteXY(SpalteMin - 1, ZeileMin - 1,
          '╔' + Doppelstrich Waagrecht + '╗');

  for Zeile := ZeileMin to  ZeileMax do
    begin
      GotoXY(SpalteMin - 4, Zeile); Write(Zeile);
      GotoXY(SpalteMin - 1, Zeile);
      Write('║', Blank String, '║');
    end;

  GotoXY(SpalteMin - 1, ZeileMax + 1);
  Write('╚', Doppelstrich Waagrecht, '╝');

  TextColor(Yellow); WriteXY(1,  25, 'F1' );
  TextColor(White);  Write('-Hilfe');
  TextColor(Yellow); WriteXY(12, 25, 'F2' );
  TextColor(White);  Write('-Setzen');
  TextColor(Yellow); WriteXY(24, 25, 'F3' );
  TextColor(White);  Write('-Löschen');
  TextColor(Yellow); WriteXY(37, 25, 'F10');
  TextColor(White);  Write('-End-Beiprogramm');

  WriteXY(59, 25, 'Spalte: ');
  WriteXY(71, 25, 'Zeile: ' );
end;  { von "procedure Bildschirmaufbau" }
```

```
procedure Hilfe;  { ---------------------------------------------- }
const
  Spalte = 15;   { lokale Konstante !! }
begin
  TextColor(Yellow);
  WriteXY(Spalte, 4, '+---------------- Hilfe ------------------+');
  WriteXY(Spalte, 5, '¦ Mit Cursor positionieren                 ¦');
  WriteXY(Spalte, 6, '¦ Mit "F2" Punkt setzen                    ¦');
  WriteXY(Spalte, 7, '¦ Mit "F3" Punkt löschen                   ¦');
  WriteXY(Spalte, 8, '¦ Mit "F10" Wechsel zum Beiprogramm, Ende ¦');
  WriteXY(Spalte, 9, '+---------------------- Zurück mit "ESC" --+');
  GotoXY(Spalte + 35, 9);
  TextColor(White);

  repeat
  until ReadKey = #27;    { "#27" Steuerzeichen Escape    }

  Legende;                { Aufruf der Prozedur "Legende" }

 { Durch nochmaliges Schreiben der Legende wird eine einfache        }
 { "Fenstertechnik" demonstriert. Sie funktioniert hier einwandfrei, }
 { weil die Zeilen der Legende immer länger sind als die Zeilen der  }
 { Hilfetexte. Im allgemeinen Fall sollte man die Teile des Bild-    }
 { schirms, die durch ein Fenster temporär überschrieben werden,     }
 { aus dem Bildschirmspeicher heraus auf eine Variable kopieren und  }
 { später wieder in den gleichen Bildschirmspeicherbereich zurück-   }
 { kopieren. Siehe dazu Kapitel 27: "Systemnahe Programmierung".     }
end;  { von Prozedur "Hilfe" }


procedure Beiprogramm; { ---------------------------------------- }
const
  FrequenzMin = 40;    { Frequenzen in Hertz   }
  FrequenzMax = 4000;  { für Prozedur "Sound"  }
  DauerMin    = 10;    { Zeit in Millisekunden }
  DauerMax    = 200;   { für Prozedur "Delay"  }

begin
  for Zeile := 1 to ZeileMin - 2 do
    begin
      GotoXY(1, Zeile);
      ClrEoL;    { Überschrift und Legende löschen }
    end;

  for Zeile := ZeileMin to ZeileMax do       { Das Innere  }
    for Spalte := SpalteMin to SpalteMax do  { des Kastens }
      WriteXY(Spalte, Zeile, ' ');           { löschen     }

  TextColor(Yellow);
  WriteXY(18,  5, 'Im Beiprogramm eine Zufallsspielerei');

  GotoXY(1,  25); ClrEoL;
  WriteXY(21, 25, 'Ende mit beliebigem Tastendruck');
  repeat
    Spalte := Random(SpalteMax - SpalteMin + 1) + SpalteMin;
    Zeile  := Random(ZeileMax  - ZeileMin  + 1) + ZeileMin;
    WriteXY(Spalte, Zeile, PunktSetzen);
    { Wer keine Nerven hat, muß die folgenden }
    { drei Zeilen löschen >>>>                }
```

```
    Sound(Random(FrequenzMax - FrequenzMin + 1) + FrequenzMin);
    Delay(Random(DauerMax    - DauerMin    + 1) + DauerMin);
    NoSound;
    Spalte := Random(SpalteMax - SpalteMin + 1) + SpalteMin;
    Zeile  := Random(ZeileMax  - ZeileMin  + 1) + ZeileMin;
    WriteXY(Spalte, Zeile, PunktLoeschen);
  until KeyPressed;
  TextColor(White);
end;    { von Prozedur "Beiprogramm" }


begin   { ============ Beginn Hauptprogramm ======================== }
  ZulaessigeZeichen := F1  +  F2   +  F3  +  F10  +
                       CursorLinks + CursorRechts +
                       CursorUnten + CursorOben;
  Bildschirmaufbau;    { Aufruf der Prozedur "Bildschirmaufbau" }

  Zeile  := (ZeileMax  + ZeileMin ) div 2;  { Für Start Cursor ... }
  Spalte := (SpalteMax + SpalteMin) div 2;  { ... in die Mitte     }

  repeat
    TextColor(Yellow);
    GotoXY(67, 25); Write(Spalte);
    GotoXY(78, 25); Write(Zeile );
    TextColor(White);

    GotoXY(Spalte, Zeile);  { bei jedem Durchlauf Cursor }
                            { neu positionieren          }
    repeat
      ErstesByte := ReadKey;
      if ErstesByte = #0
         then ZweitesByte := ReadKey;
    until (ErstesByte = #0) and
          (Pos(ZweitesByte, ZulaessigeZeichen) <> 0);

    case ZweitesByte of
      CursorLinks:  Dec(Spalte);  { Spalte - 1}
      CursorRechts: Inc(Spalte);  { Spalte + 1}
      CursorUnten:  Inc(Zeile );  { Zeile  + 1}
      CursorOben:   Dec(Zeile );  { Zeile  - 1}
      F1:           Hilfe;        { Aufruf der Prozedur "Hilfe" }
      F2:           begin
                      Write(PunktSetzen);
                      Inc(Spalte);
                    end;
      F3:           begin
                      Write(PunktLoeschen);
                      Inc(Spalte);
                    end;
    end;   { von "case ZweitesByte of" }

    if Spalte > SpalteMax then
       begin
         Spalte := SpalteMin;
         Inc(Zeile);                    { Nur akustische }
         Sound(440); Delay(10); NoSound;  { Spielerei !!!  }
       end;
    if Spalte < SpalteMin then
       begin
         Spalte := SpalteMax;
         Dec(Zeile);
```

```
        Sound(2000); Delay(10);  NoSound;
      end;
    if Zeile > ZeileMax then
      begin
        Zeile := ZeileMin;
        Sound(1000); Delay(200); NoSound;
      end;
    if Zeile < ZeileMin then
      begin
        Zeile := ZeileMax;
        Sound(4000); Delay(200); NoSound;
      end;
  until ZweitesByte = F10;

  Beiprogramm;  { Aufruf der Prozedur "Beiprogramm"  }
end.
```

22.4 Demo-Programm 3: Ein einfacher Zeileneditor

```
program Pas22041;  { "Z-Edit.PAS". Einfacher Zeileneditor }
                   { Dr. K. Haller, FH München, Stg DR    }
uses
  CRT;

var
  EingabeStr: string;


procedure Zeileneditor(var EingabeStr: string); {-------------------+}
var
  Ch: Char;
begin
  while KeyPressed do { Tastaturpuffer leeren }
    Ch := ReadKey;
  ;
  EingabeStr := '';
  repeat
    Ch := ReadKey;
    if Ch = #00 then          { Bei Sondertasten das zweite    }
      begin                   { Byte "verbraten", ausgenommen  }
        Ch := ReadKey;        { "Cursor links" = #75 = "K",    }
        if Ch = #75           { das wie #08 = BS = Backspace   }
          then Ch := #08      { behandelt werden soll.         }
          else Ch := #00;     { Zeichen #00 hier nur als Dummy }
      end;
    ;
    case Ch of
    #00: ;  { Tue bei #00 nichts, nur Dummy von vorhin }
    #27: begin  { #27 = Escape }
           EingabeStr := #27;
           Break;
         end;
    #13: Break; { #13 = Carriage Return }
    #08: if Length(EingabeStr) >= 1 then
```

```
            begin
              Write(#08, ' ', #08);  { #08 = Backspace }
              EingabeStr := Copy(EingabeStr, 1,
                                 Length(EingabeStr) - 1);
            end;
    else begin
           EingabeStr := EingabeStr + Ch;
           Write(Ch);
         end;
    end;
  until Ch = #13;
  ;
  while EingabeStr[1] = ' ' do  { Führende Blanks entfernen.  }
    if EingabeStr[0] = #00      { "Break" notwendig, wenn der }
       then Break               { String nur Blanks enthält!  }
       else EingabeStr := Copy(EingabeStr, 2, Length(EingabeStr) - 1);
  while EingabeStr[Length(EingabeStr)] = ' ' do { Nachstehende Blanks}
    EingabeStr := Copy(EingabeStr, 1, Length(EingabeStr) - 1); { weg }
end; {-------------------------------------------------------------+}


begin {---------------------- Demo-Main ----------------------------+}
  repeat
    TextBackGround(Blue); TextColor(Yellow); ClrScr;
    GotoXY(15,  6); Write('Ende mit Taste "Enter" oder ' +
                          'Leerzeichen-String');
    TextColor(White);
    GotoXY(15, 10); Write('Eingabe: ¦');
    TextColor(Yellow);
    ;
    Zeileneditor(EingabeStr);
    ;
    TextColor(White);
    GotoXY(15, 11); Write('Ausgabe: ¦');
    TextColor(Yellow);
    Write(EingabeStr);
    TextColor(White); Write('¦');
    GotoXY(15, 13); Write('Ende mit Tastendruck ... ');
    repeat
    until ReadKey <> '';
  until EingabeStr = '';
end.  {-------------------------------------------------------------+}
```