```
////////////////////////////////////////////////////////////////////////////
//  Projektname : Mastermind
//   Autor: Alexander Hofstätter
//
//   Aufgabe :  Der Computer erzeugt unsichtbar einen zufälligen vierstelligen Code
//   aus den 4 Farben (rot,blau,grün,schwarz).
//   Man soll durch geschicktes Kombinieren dann versuchen diesen Code zu knacken.
//   Zu jedem richtigen Codezeichen wird eine Erfolgsmeldung angezeigt -
//    *  : für richtige Farbe am falschen Platz.
//   +  : für richtige Farbe am richtigen Platz.
//
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
// ----------- Header-Dateien ---------------------------------------------//
//                                                                          //
////////////////////////////////////////////////////////////////////////////////

 #include <stdio.h>
 #include <stdlib.h>
 #include <iostream>
 #include <time.h>
 #include "conio2.h"
 #include <string>
 #include <windows.h>


using namespace std;

////////////////////////////////////////////////////////////////////////////////
// ----------- Ersatzwerte definieren -------------------------------------//
//                                                                          //
////////////////////////////////////////////////////////////////////////////////
 #define ESC        27
 #define BLOCK      219
 #define ENTER      13
 #define LINKS      75
 #define RECHTS     77
 #define RUNTER     80
 #define RAUF       72
 #define BACKSPACE  8
 #define SPACE      32
 #define SCHWARZ    0
 #define BLAU       1
 #define GRUEN      2
 #define ROT        4
 #define FALSE      0
 #define TRUE       1

////////////////////////////////////////////////////////////////////////////////
// ----------- globale Variablen definieren -------------------------------//
//                                                                          //
////////////////////////////////////////////////////////////////////////////////

 int felder_max = 4;   // Anzahl der Farbfelder
 int farben_max = 4;   // Anzahl der Farben
 int runden_max = 12;  // Anzahl der maximalen runden
 int feld_hoehe = 3;
 int feld_breite = 5;
 int startx,endx;
 const int hintergrund = DARKGRAY;
 const int menuefarbe = BLUE;
 const int fenster_breite = 150;
 const int fenster_hoehe = 250;


////////////////////////////////////////////////////////////////////////////////
/** ----------- Funktionsdefinitionen ---------------------------------*/
//                                                                          //
////////////////////////////////////////////////////////////////////////////////

/**---------- Funktion farbewechseln ---------------------------------------*
 *   ... Wechselt die Farbe eines Feldes nach drücken der Leertaste um eine *
 *   Position. Das Farbefeld wird über die momentane x Position ermittelt.  *
```

```
 *-------------------------------------------------------------------------*/
void farbewechseln( int farbe[], int x, int x1, int x2)
{   for ( int i = 0; i < farben_max; i++)
    {   if (x == x1 +  i * x2)                          // aktuelle X-Position gleich dem Anfangswert von x + i mal abstand
        {   if (farbe[i] >= farben_max)  farbe[i] = 0;  // Wenn die letzte Farbe erreicht ist, springe zur 1.ten Farbe
            else farbe[i]++;                           // Sonst ( farbe < farben_max && farbe >= 0 ), nächste Farbe
        }
        if (farbe[i] == hintergrund)                    // Hintergrundfarbe überspringen
            farbe[i]++;
        if (farbe[i] == CYAN && farben_max == 4)        // Cyan bei 4 Farben überspringen
            farbe[i]++;
    }
}

/**---------- Funktion farbausgabe -------------------------------------------*
 *    ... Gibt alle Farben eines Arrays in 3*5 Blöcken aus.                  *
 *    Dies wird für alle felder_max Farbpositionen gemacht                   *
 *-------------------------------------------------------------------------*/
void farbausgabe ( int farbe[], int y, int xabstand )
{   int x = startx - feld_breite/2;
    for ( int i = 0 ; i < felder_max; i++, x += xabstand)
    {   for ( int y1 = -feld_hoehe/2; y1 <= feld_hoehe/2; y1++) // Cursor vertikale mittig setzen
        {   textcolor(farbe[i]);                               // Setze Zeichenfarbe vom Array
            gotoxy(x,y+y1);                                    // Cursor platzieren
            for ( int l=1; l<=feld_breite; l++)                // Ganze Breite ausfüllen
            {   printf("%c",BLOCK);                           // Ascii Zeichen 219 ausgeben
            }
        }
    }
}

/**---------- Funktion zufallsfarbe ------------------------------------------*
 *   ... Weist dem komplettem array eine zufällige Zahl (Farbe) von 0 bis   *
 *   farben_max zu. Dies wird für alle felder_max Farbpositionen gemacht    *
 *-------------------------------------------------------------------------*/
void zufallsfarbe ( int farbe[] )
{   for( int i = 0; i < felder_max; i++)            // Array durchlaufen
    {   farbe[i] = rand()% farben_max;              // Zufallszahl von 0 bis farben_max
        if (farbe[i] == hintergrund)
            farbe[i]++;                             // Hintergrundfarbe überspringen
        if (farbe[i] == CYAN && farben_max == 4)    // Cyan bei 4 Farben überspringen
            farbe[i]++;
        for(int j = 0; j < i; j++)                  // Alle bisherigen Farben durchlaufen
        {   if(farbe[j] == farbe[i])                // Prüfe ob Farbe schon vorhanen ist
            {   i--; break;                         // Schleife abbrechen, i nochmal durchgehen
            }
        }
    }
}

/**---------- Funktion trennlinie --------------------------------------------*
 *   ... Gibt eine Trennlinie mit wählbarer Länge, Farbe, wählbaren         *
 *   x und y Koordinaten und einem wählbarem Zeichen aus                    *
 *-------------------------------------------------------------------------*/
void trennlinie (int laenge, int x, int y, char zeichen, int zeilen, int color)
{   textcolor(color);                       // gewählte Farbe setzen
    for (int i=0;i<zeilen;i++)              // durchläuft die Zeilen
    {   gotoxy((x-laenge)/2,y+i);           // zentriert die Trennlinie
        for (int j=0;j<laenge;j++)          // Pro Zeile, die volle Länge ausgeben
            printf("%c",zeichen);           // Zeichen ausgeben
    }
    textcolor(menuefarbe);                  // Die Farbe zurück auf die Menüfarbe setzen
}

/**---------- Funktion array_init --------------------------------------------*
 *    ... Initialisierung eines Arrays mit einem gewünschten Wert          *
 *-------------------------------------------------------------------------*/
void array_init ( int array[], int wert, int max )
{   for ( int i=0; i<max; i++)   // durchläuft das ganze Array
    {   array[i]=wert;           // und weißt jedem Feldelement einen Wert zu
    }
}

/**---------- Funktion einstellungen -----------------------------------------*
 *   ... Lest neue Werte für beliebige Variablen ein und ändert diese       *
```

```
 *   wenn sie den Vorgaben entsprechen                                      *
 *-------------------------------------------------------------------------*/
int einstellungen (int ypos, int starty, string menue_name, string fehler, int max_wert, int wert)
{   int zahl;
    do
    {   zahl = 0;                                          // Hilfsvariable initialisieren
        gotoxy(30,starty);
        printf(">> Bearbeitungsmodus\n\n");                // Bearbeitungsmodus in rot ausgeben
        gotoxy(7,starty+1+ypos);                           // Cursor auf passendem Menüeintrag platzieren
        printf ("[%d] ... %-20s  ...   %4d   neuer Wert: ",ypos, menue_name.c_str(), wert);     // Menüeintrag in rot überschreiben
        scanf("%d",&zahl);                                 // Neuen Wert einlesen
        if (zahl <= max_wert)return zahl;                  // Wenn der Wert den Vorgaben entspricht, dann Wert zurückgeben und abbrechen
        gotoxy(2,25);                                      // Cursor am unteren Fensterrand platzieren
        printf("%s",fehler.c_str());                       // Wenn nicht, dann passenden Fehler ausgeben
    }while(zahl > max_wert);                               // Solange wiederholen bis der Wert passt
    return zahl;
}

/**---------- Funktion setze_hintergrund ----------------------------------*
 *    ... Überschreibt das komplette Fenster mit einem neuem Hintergrund    *
 *--------------------------------------------------------------------------*/
void setze_hintergrund(int x, int y, int color)
{   textbackground(color);          // Setz den neuen Hintergrund
    for (int i=0;i<y;i++)           // Fensterbreite durchlaufen
        for (int j=0;j<x/20;j++)    // Fensterhöhe durchlaufen
            printf("                ");
    gotoxy(1,1);                    // Positioniert den Cursor auf 1|1
}

/**---------- Funktion goto_printf ----------------------------------------*
 *    ... Platziert den Cursor und gibt über printf einen string aus        *
 *--------------------------------------------------------------------------*/
void goto_printf(int x, int y, string ausgabe)
{   gotoxy(x,y);
    printf ("%s",ausgabe.c_str());
}

/**---------- Funktion menue_start ----------------------------------------*
 *    ... erste Funktion in jedem Menü. Löscht den Bildschirm, setz         *
 *    die Textfarbe, gibt einen ascii-art-Text aus und gibt die aktuelle    *
 *    Y-Position als Rückgabewert zurück                                    *
 *--------------------------------------------------------------------------*/
int menue_start(int art_color, int breite, int y, string menue_name)
{   textbackground(hintergrund);        // Setzt die Hintergrundfarbe
    clrscr();
    textcolor(art_color);               // Setzt die Textfarbe
    gotoxy(1,1);
    int x = breite/2-35;                // Berechnet die Mitte für den Text
    goto_printf(x,wherey(),"    MMMM     MMMM      AAA      SSSSSSSS   TTTTTTTTTTT  EEEEEEE  RRRRR   \n");
    goto_printf(x,wherey(),"    MM MM   MM MM     AA AA    SSSS           TTT       EE       RR  RR  \n");
    goto_printf(x,wherey(),"    MM  MM MM  MM    AAAAAAA    SSSSSSSS      TTT       EEEEEEE  RRRRR   \n");
    goto_printf(x,wherey(),"    MM   MMM   MM   AA     AA        SSSS     TTT       EE       RR  RR  \n");
    goto_printf(x,wherey(),"    MM    M    MM  AA       AA  SSSSSSSS      TTT       EEEEEEE  RR   RR \n\n");

    goto_printf(x,wherey(),"              MMMM     MMMM  IIIIIIIIIII  NNNN    NN  DDDDD    \n");
    goto_printf(x,wherey(),"              MM MM   MM MM      III      NN NN   NN  DD  DDD  \n");
    goto_printf(x,wherey(),"              MM  MM MM  MM      III      NN  NN  NN  DD   DDD \n");
    goto_printf(x,wherey(),"              MM   MMM   MM      III      NN   NN NN  DD  DDD  \n");
    goto_printf(x,wherey(),"              MM    M    MM  IIIIIIIIIII  NN    NNNN  DDDDDD \n\n\n\n");
    textcolor(menuefarbe);
    printf ("  %s\n\n",menue_name.c_str());
    return wherey()-2;                  // Gibt die aktuelle Y-Position als Rückgabewert zurück

}

/**---------- Funktion vergleichen ----------------------------------------*
 *    ... der Vergleichs - algorithmus prüft ob es direkte oder indirekte   *
 *    Treffer gibt und gibt die passenden Symbole dazu aus                  *
 *--------------------------------------------------------------------------*/
int vergleichen(int array_1[], int array_2[], int xpos, int ypos)
{   int direkt = 0, indirekt = 0;       // lokale Zählvariablen für direkte und indirekte Treffer
    for(int i = 0; i < felder_max; i++)
        for(int j = 0; j < felder_max; j++)
            if (array_1[i] == array_2[j])       // Prüfe ob irgendeine Farbe des array_2 gleich einer Farbe des array_1 ist
            {   if (array_1[i] == array_2[i])   // Wenn beide Farben den gleichen Index haben -> direkter Treffer
```

```
                {   direkt++; break;
                }
                else
                {   indirekt++; break;          // Bei zwei gleichen Farben mit unterschiedlichem Index  -> indirekter Treffer
                }
            }
    gotoxy(xpos+2,ypos);                          // Positioniere Cursor 2 Zeichen entfernt vom rechten äußerem Feld
    textcolor(SCHWARZ);                          // Setze Zeichenfarbe
    for (int i = 0 ; i < direkt; i++)
        printf("%2c",'+');      // Gibt das Zeichen '+' für jeden direkten Treffer in schwarz aus
    textcolor(WHITE);
    for (int i = 0 ; i < indirekt; i++)
        printf("%2c",'*');   // Gibt das Zeichen '*' für jeden indirekten Treffer in weiß aus
    return direkt;
}

/**---------- Funktion datei ------------------------------------------------*
 *    ... öffnet eine Datei zum lesen und gibt sie 1:1 am Bildschrim aus    *
 *--------------------------------------------------------------------------*/
int datei(char *dateiname)
{   FILE *datei;                            // Zeiger auf datei
    char zeichen;                           // char Variable um die Zeichen auszugeben
    datei = fopen(dateiname,"r");           // Datei zum Lesen öffnen
    if(datei==NULL)                         // Fehlerbehandlung
        return 1;
    while ((zeichen=fgetc(datei))!=EOF)     // zeichweise lesen bis Dateiende erreicht ist
    {   putchar(zeichen);                   // Zeichen am Bildschirm ausgeben
    }
    fclose(datei);                          // Datei schließen
    return 0;
}

////////////////////////////////////////////////////////////////////////////
/** ----------- Hauptptogramm ------------------------------------------------*/
//                                                                          //
////////////////////////////////////////////////////////////////////////////
int main()
{
/*   Zeichen     Hex    Okt
    ========================
       'Ä'       8E     216
       'ä'       84     204
       'Ö'       99     231
       'ö'       94     224
       'Ü'       9A     232
       'ü'       81     201
       'ß'       E1     341   */

    /**---------- lokale Variablen ---------------------------------------------*/
    int code[felder_max];       // Feld für den zufälligen generierten Farbcode
    int erraten[felder_max];    // Feld für die vom Spieler eingegeben Lösung
    int runde;                  // Zählvariable für die momentane Runde
    int xpos,ypos;              // Variable für die momentane X- und Y- Position
    int abstand;                // Variable für den Abstand zwischen der Mitte eines Farbfeldes zur nächsten Mitte des nächsten Farbfeldes
    bool aktives_spiel = FALSE; // Variable für logische Werte True und False für die Abfrage ob gespielt wird oder nicht
    char key;                   // Variable für diverse Menüabfragen

    srand(time(NULL));                                              // Zufallsfolge initialisieren
    _setcursortype(100);                                            // Cursor auf volle Größe einstellen
    SetConsoleTitle(" Mastermind :: Hack the Code, \270 2012 by A. Hofst\204tter");   // Konsolen Titel setzen
    do
    {   if (aktives_spiel == FALSE)                                 // Wenn das Spiel nicht läuft kommt man ins Menü
        {   /**---------- Menübereiche -------------------------------------------*/
            system("mode con cols=80 lines=25");                    // Konsolengröße unter Windows auf 80 x 25 skalieren
            const int starty = menue_start(SCHWARZ,80,starty,"Startmen\201");   // Konstante Variable für die vertikale Startposition
            printf ("      [1]   ... Neues Spiel starten\n");
            printf ("      [2]   ... Modus w\204hlen\n");
            printf ("      [3]   ... Spielregeln\n");
            printf ("      [4]   ... Einstellungen\n");
            printf ("      [ESC] ... Programmende     %d\n",farben_max);

            key = getch();
            if (key == '1')
            {   /**---------- Neues Spiel starten ------------------------------------*/
                system("mode con cols=150 lines=250");          // Konsolengröße unter Windows auf 150 x 250 für das Menü skalieren
```

```
301                 setze_hintergrund(150,250,hintergrund);         /**kompletten*/// Konsolenhintergrund setzen
302
303                 /*---------- Berechnungen für X- und Y- Positionen ------------------------*/
304                 abstand = feld_breite + 2;
305                 xpos = startx = fenster_breite/2 - (felder_max * abstand)/2 + abstand/2;
306                 endx = (felder_max - 1) * abstand + startx;
307                 ypos = wherey() + feld_hoehe + 2;
308
309                 array_init(code,SCHWARZ,felder_max);            // Initialisierung des Arrays code bis felder_max
310                 array_init(erraten,SCHWARZ,felder_max);
311                 zufallsfarbe(code);                             // Erzeugt einen zufälligen Farbcode für das Array
312                 aktives_spiel = TRUE;                           // aktives Spiel auf True setzen
313                 runde = 1;                                      // und die Runde auf 1
314             }
315
316             if (key == '2')
317             {   /**---------- Modus wählen -------------------------------------------------*/
318                 do
319                 {   menue_start(SCHWARZ,80,starty,"Startmen\201 >> Modus w\204hlen");
320                     printf ("      [1] ... Mastermind easy   (4 Felder, 4 Farben) \n");   // Mastermind easy = Standardeinstellung
321                     printf ("      [2] ... Mastermind        (4 Felder, 6 Farben) \n");
322                     printf ("      [3] ... Super Mastermind  (5 Felder, 8 Farben) \n");
323                     printf ("      [4] ... Benutzerdefinierter Modus  \n");
324                     printf ("      [<] ... Zur\201ck ins Hauptmen\201 \n");
325
326                     /*---------- Prüft welche der o.g. Option zur Zeit aktiv ist, -------------*/
327                     /*---------- und gibt ein rotes "aktiviert" neben der aktiven Option aus --*/
328                     textcolor(ROT);
329                     if      (felder_max == 4 && farben_max == 4 )
330                         goto_printf(58,starty+1+1,"...     aktiviert");
331                     else if (felder_max == 4 && farben_max == 6 )
332                         goto_printf(58,starty+1+2,"...     aktiviert");
333                     else if (felder_max == 5 && farben_max == 8 )
334                         goto_printf(58,starty+1+3,"...     aktiviert");
335                     else
336                         goto_printf(58,starty+1+4,"...     aktiviert");
337
338
339                     /*---------- Ändert die jeweiligen Werte ----------------------------------*/
340                     key=getch();                // fragt die Taste ab
341                     if (key == '1')
342                     {   farben_max = 4; felder_max = 4;         // Setze auf Mastermind easy
343                     }
344                     if (key == '2')
345                     {   farben_max = 6; felder_max = 4;            // Setze auf Mastermind
346                     }
347                     if (key == '3')
348                     {   farben_max = 8; felder_max = 5;         // Setze auf Super Mastermind
349                     }
350                     if (key == '4')                 // Benutzerdefinierter Modus
351                     {   goto_printf(2,24,"Die Einstellungen f\201r den benutzerdefinierten Modus m\201ssen unter");
352                         goto_printf(2,25,"Startmen\201 >> Einstellungen angepasst werden");
353                         getch();        // Warte auf Tastendruck
354                     }
355                     textcolor(menuefarbe);
356                 }while(key != BACKSPACE && key != LINKS && key != ENTER);
357             }
358
359             if (key == '3')
360             {   /**---------- Spielregeln --------------------------------------------------*/
361                 menue_start(SCHWARZ,80,starty,"Startmen\201 >> Regeln");
362                 datei( (char*) "dat/rules.ini");        // Typumformung von string zu 'char*'
363                 getch();
364             }
365
366             if (key == '4')
367             {   /**---------- Einstellungen ------------------------------------------------*/
368                 do
369                 {   menue_start(SCHWARZ,80,starty,"Startmen\201 >> Einstellungen");
370                     printf ("      [1] ... Anzahl an Farben      ...   %4d\n",farben_max);
371                     printf ("      [2] ... Anzahl an Feldern     ...   %4d\n",felder_max);
372                     printf ("      [3] ... Maximale Runden       ...   %4d\n",runden_max);
373                     printf ("      [4] ... H\224he eines Feldes     ...   %4d\n",feld_hoehe);
374                     printf ("      [5] ... Breite eines Feldes   ...   %4d\n",feld_breite);
375                     printf ("      [<] ... Zur\201ck ins Hauptmen\201 \n");
```

```
                    key=getch();
                    textcolor(ROT);
                    if (key == '1')
                        farben_max = einstellungen (1,starty,"Anzahl an Farben","Fehler: Es stehen nur 16 Farben zur Verf\201gung !",16,farben_max);
                    if (key == '2')
                        felder_max = einstellungen (2,starty,"Anzahl an Feldern","Fehler: Die Anzahl der Felder muss kleiner/ gleich der Anzahl der Farben sein!",felder_max,felder_max);
                    if (key == '3')
                        runden_max = einstellungen (3,starty,"Maximale Runden","Fehler: Mehr als 40 Runden sind leider nicht m\224glich!",40,runden_max);
                    if (key == '4')
                        feld_hoehe = einstellungen (4,starty,"H\224he eines Feldes","Fehler: Ein Farbfeld kann nicht h\224her als 15 Zeichen sein!",15,feld_hoehe);
                    if (key == '5')
                        feld_breite = einstellungen (5,starty,"Breite eines Feldes","Fehler: Ein Farbfeld kann nicht breiter als 20 Zeichen sein!",20,feld_breite);

                    /*---------- Berechnungen für X- und Y- Positionen erneut ausführen -------*/
                    abstand = feld_breite+2;
                    startx = fenster_breite/2 - (felder_max * abstand)/2 + abstand/2;
                    endx = (felder_max-1) * abstand + startx;

                    /*---------- Prüfen, ob die insgesamte Breite noch ins Fenster passt ------*/
                    if (feld_breite * felder_max >= fenster_breite)
                    {   while(feld_breite * felder_max >= fenster_breite)
                        {   feld_breite--;      // Wenn nicht, dann feld_breite solange vermindern bis es passt
                            gotoxy(2,25);
                            printf(" Um Konflikte zu vermieden, wurde die Breite des Feldes auf %d reduziert! ",feld_breite);
                        }
                        textcolor(menuefarbe);
                        getch();
                    }

                }while(key != BACKSPACE && key != LINKS && key != ENTER);
            }
        }
        else
        {   /**---------- eigentliches Spiel ----------------------------------------*/
            if (runde <= runden_max)                                    // Solange maximal nicht Runden überschritten
                farbausgabe(erraten,ypos,abstand);                      // Nächste Ratelösung vom Spieler ausgeben
            gotoxy(xpos,ypos);                                          // Zur letzten X- und Y- Position springen
            key = getch();
            if (key == ENTER)                                           // Option ENTER (Ratelösung wird abgeschickt und verglichen)
            {   /**---------- Vergleichsalgorithmus --------------------------------------*/
                int direkt = vergleichen(code,erraten,endx+feld_breite/2,ypos);

                /*----------- Gewinnabfrage ---------------------------------------------*/
                if (runde == runden_max || direkt == felder_max)        // maximal Anzahl an Runden erreicht, oder Code geknackt?
                {   trennlinie(endx-startx+15,fenster_breite,ypos + feld_hoehe - feld_hoehe/4,'=',2,LIGHTRED);  // 2 Trennlinien ausgeben
                    farbausgabe(code,ypos + feld_hoehe + feld_hoehe + 2,abstand);      // Computerlösung ausgeben zur Kontrolle
                    ypos = wherey() + feld_hoehe;                       // Ypos um die Höhe eines Feldes erhöhen
                    textcolor(menuefarbe);

                    if (direkt == felder_max)
                    {   gotoxy(fenster_breite/2-25,ypos);
                        printf("Gl\201ckwunsch, du hast das Spiel in %d Runden gewonnen\n",runde);
                    }
                    else
                        goto_printf(fenster_breite/2-35,ypos,"Du hast leider verloren! Du solltest den Schwierigkeitsgrad verringern!");

                    trennlinie(endx-startx+15,fenster_breite,ypos+2,'=',1,LIGHTRED);
                    aktives_spiel = FALSE;                              // aktives Spiel auf FALSE setzen
                    goto_printf(fenster_breite/2-16,ypos+4," [1]    ...  Nochmal spielen\n");
                    goto_printf(fenster_breite/2-16,ypos+5," [<]    ...  Zur\201ck ins Hauptmen\201\n");
                    goto_printf(fenster_breite/2-16,ypos+6," [ESC]  ...  Programmende\n");

                    key = getch();                                      // Neues Spiel starten
                    if (key == '1')
                    {   /*----------- Neues Spiel starten ----------------------------------------*/
                        system("mode con cols=150 lines=250");          // Konsolengröße unter Windows auf 150 x 250 für das Menü skalieren
                        setze_hintergrund(150,250,hintergrund);         // kompletten Konsolenhintergrund setzen

                        /*---------- Berechnungen für X- und Y- Positionen ------------------------*/
                        abstand = feld_breite + 2;
                        xpos = startx = fenster_breite/2 - (felder_max * abstand)/2 + abstand/2;
                        endx = (felder_max - 1) * abstand + startx;
                        ypos = wherey() + feld_hoehe + 2;

                        array_init(code,SCHWARZ,felder_max);            // Initialisierung des Arrays code bis felder_max
```

```
                        array_init(erraten,SCHWARZ,felder_max);
                        zufallsfarbe(code);                             // Erzeugt einen zufälligen Farbcode für das Array
                        aktives_spiel = TRUE;                           // aktives Spiel auf True setzen
                        runde = 1;
                    }
                }
                else if (runde < runden_max)                            // Wenn noch nicht gewonnen, runde und Y-Position erhöhen
                {   runde++;
                    ypos += feld_hoehe + 2;
                }
            }
            gotoxy(xpos,ypos);
            /*----------- Linke Pfeiltaste: X-Position verringern ---------------------*/
            if (key == LINKS  && wherex() > startx)   gotoxy(wherex()-abstand,wherey());
            /*----------- Rechte Pfeiltaste: X-Position vergrößern --------------------*/
            if (key == RECHTS && wherex() <   endx)   gotoxy(wherex()+abstand,wherey());
            /*----------- Leertaste: Farbe des aktuellen Feldes ändern ----------------*/
            if (key == SPACE)   farbewechseln(erraten, wherex(), startx, abstand );
            xpos=wherex();                                              // X-Position merken
            ypos=wherey();
        }
    }while(key!=ESC);                                                   // Abbruch bei ESC Taste
    return 0;
}

```