Lösungen zur Vorlesung
GRUNDLAGEN DER INFORMATIK I
Studiengang Elektrotechnik
SS 03

AG Betriebssysteme FB3
Kirsten Berkenkötter

Lösung Übungszettel 5

1 Aufgabe 1: Tic-Tac-Toe

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/*----------------------------------------------------------------------------*/
/* Funktion zum Zeichnen des Spielfeld
 * Parameter: char array[3][3], symbolisiert das Tic-Tac-Toe-Feld
 *            Zeichen O symbolisiert Spielfigur Computer
 *            Zeichen X symbolisiert Spielfigur Mensch
 *            Leerzeichen bedeutet noch nicht gesetzt
 * Rückgabewert: keiner
 */
/*----------------------------------------------------------------------------*/

void spielfeld(char array[3][3])
{
    //Spielfeld mit Belegung malen
    printf("\n    1   2   3 \n");
    printf("   --- --- --- \n");
    printf("1 | %c | %c | %c |\n", array[0][0], array[0][1], array[0][2]);
    printf("   --- --- --- \n");
    printf("2 | %c | %c | %c |\n", array[1][0], array[1][1], array[1][2]
);
    printf("   --- --- --- \n");
    printf("3 | %c | %c | %c |\n", array[2][0], array[2][1], array[2][2]);
    printf("   --- --- --- \n\n");
}

/*----------------------------------------------------------------------------*/
/* Funktion, die ueberprueft, ob gerade jemand gewinnen kann,
 * d.h. ob in einer Spalte, Zeile oder Diagonalen zwei gleiche Zeichen
und
 * ein Leerzeichen sind
```

```
 * wenn ein Gewinn moeglich ist, werden zeile und spalte mit dem zum
Gewinn
 * zu setzenden Wert ueberschrieben und eine 0 zurueckgegeben
 * wenn kein Gewinn moeglich ist, wird -1 zurueckgegeben
 * Parameter: char array[3][3], das Spielfeld
 *            int *zeile, Zeiger auf Zeile
 *            int *spalte, Zeiger auf Spalte
 *            char zeichen, X, wenn geguckt wird, ob Mensch gewinnen
kann
 *                          O, wenn geguckt wird, ob Computer gewinnt
 * Rueckgabewert: 0, wenn Gewinn moeglich, sonst -1
 */
/*--------------------------------------------------------------------------*/

int kannGewinnen(char array[3][3], int *zeile, int *spalte, char zeichen)
{
    int i, j;

    //checke Zeilen
    for (i = 0; i < 3; i++)
    {
        //sind das erste und zweite belegt und das dritte frei?
        if((array[i][0] == zeichen) && (array[i][1] == zeichen) &&
           (array[i][2] == ' '))
        {
            *zeile = i;
            *spalte = 2;
            return 0;
        }
        //sind das erste und dritte belegt und das zweite frei?
        if((array[i][0] == zeichen) && (array[i][2] == zeichen) &&
           (array[i][1] == ' '))
        {
            *zeile = i;
            *spalte = 1;
            return 0;
        }

        //sind das zweite und dritte belegt und das erste frei?
        if((array[i][1] ==  zeichen) && (array[i][2] ==  zeichen) &&
           (array[i][0] == ' '))
        {
            *zeile = i;
```

```
            *spalte = 0;
            return 0;
        }
    }

    //checke Spalten
    for (i = 0; i < 3; i++)
    {
        //sind das erste und zweite belegt und das dritte frei?
        if((array[0][i] ==  zeichen) && (array[1][i] ==  zeichen) &&
           (array[2][i] == ' '))
        {
            *zeile = 2;
            *spalte = i;
            return 0;
        }
        //sind das erste und dritte belegt und das zweite frei?
        if((array[0][i] ==  zeichen) && (array[2][i] ==  zeichen) &&
           (array[1][i] == ' '))
        {
            *zeile = 1;
            *spalte = i;
            return 0;
        }
        //sind das zweite und dritte belegt und das erste frei?
        if((array[1][i] ==  zeichen) && (array[2][i] ==  zeichen) &&
           (array[0][i] == ' '))
        {
            *zeile = 0;
            *spalte = i;
            return 0;
        }
    }

    //checke erste Diagonale (von links oben nach rechts unten)
    //sind das erste und zweite belegt und das dritte frei?
    if ((array[0][0] ==  zeichen) && (array[1][1] ==  zeichen) &&
        (array[2][2] == ' '))
    {
        *zeile = 2;
        *spalte = 2;
        return 0;
```

```
}
//sind das erste und dritte belegt und das zweite frei?
if ((array[0][0] ==  zeichen) && (array[2][2] ==  zeichen) &&
    (array[1][1] == ' '))
{
    *zeile = 1;
    *spalte = 1;
    return 0;
}

//sind das zweite und dritte belegt und das erste frei?
if ((array[1][1] ==  zeichen) && (array[2][2] ==  zeichen) &&
    (array[0][0] == ' '))
{
    *zeile = 0;
    *spalte = 0;
    return 0;
}

//checke zweite Diagonale (von rechts oben nach links unten)
//sind das erste und zweite belegt und das dritte frei?
if ((array[0][2] ==  zeichen) && (array[1][1] ==  zeichen) &&
    (array[2][0] == ' '))
{
    *zeile = 2;
    *spalte = 0;
    return 0;
}

//sind das erste und dritte belegt und das zweite frei?
if ((array[0][2] ==  zeichen) && (array[2][0] ==  zeichen) &&
    (array[1][1] == ' '))
{
    *zeile = 1;
    *spalte = 1;
    return 0;
}

//sind das zweite und dritte belegt und das erste frei?
if ((array[1][1] ==  zeichen) && (array[2][0] ==  zeichen) &&
    (array[0][2] == ' '))
{
```

```
        *zeile = 0;
        *spalte = 2;
        return 0;
    }

    //im Moment kein Gewinn moeglichn, Rueckgabewert -1
    return -1;
}

/*----------------------------------------------------------------------------*/
/* Funktion, die den Spielzug des Computers uebernimmt
 * Parameter: charr array[3][3], das Spielfeld
 * Rueckgabewert: keiner
 */
/*----------------------------------------------------------------------------*/

void spielzugComputer(char array[3][3])
{
    int nok, zeile, spalte;

    //fuer die Zufallszahlen
    srand(time(NULL));

    //zunaechst wird ueberprueft, ob der Computer gewinnen kann
    //wenn dies der Fall ist, braucht der Spielzug nicht durch
    //Zufallszahlen ermittelt werden, sondern ist durch zeile
    //und spalte bestimmt, damit der Computer gewinnt
    //ausserdem wird ueberprueft, ob der Mensch gewinnen kann
    //wenn dies der Fall ist, braucht der Spielzug nicht durch
    //Zufallszahlen ermittelt werden, sonder ist durch zeile
    //und spalte bestimmt, damit der Computer nicht verliert
    //wird fuer beide Ueberpruefungen -1 zurueckgegeben, kann
    //gerade keiner gewinnen und der Spielzug wird durch
    //Zufallszahlen bestimmt
    if((kannGewinnen(array, &zeile, &spalte, '0') == -1) &&
       (kannGewinnen(array, &zeile, &spalte, 'X') == -1))
    {
        do
        {
            //Zufallszahlen ermitteln, bis ein gueltiges Feld
            //gefunden wird
            nok = 0;
```

```
            zeile = rand()%3;
            spalte = rand()%3;

            //Feld schon belegt?
            if ((array[zeile][spalte] == 'X') ||
                (array[zeile][spalte] == 'O'))
            {
                nok = 1;
            }
        } while (nok);
    }

    //Feld mit Spielfigur 0 markieren
    array[zeile][spalte] = 'O';
}

/*-------------------------------------------------------------------------*/
/* Funktion, um den Menschen seinen Zug machen zu lassen
 * Parameter: das Spielfeld
 * Rueckgabewert: keiner
 */
/*-------------------------------------------------------------------------*/

void spielzugMensch(char array[3][3])
{
    int nok, zeile, spalte;

    //solange abfragen, bis ein gueltiges Feld eingegeben wurde
    do
    {
        //Feld gueltig
        nok = 0;

        //Zeile abfragen
        printf("Welche Zeile moechten Sie setzen?");
        scanf("%d", &zeile);

        //Spalte abfragen
        printf("Welche Spalte moechten Sie setzen?");
        scanf("%d", &spalte);
```

```
        //Zeile und Spalte muss zwischen 1 und 3 liegen
        if ((zeile < 1) || (zeile > 3) || (spalte < 1) || (spalte >
3))
        {
            //Feld ungueltig
            nok = 1;
            printf("Falsche Eingabe!\n");
        }

        //Feld schon belegt?
        if ((array[zeile-1][spalte-1] == 'X') ||
            (array[zeile-1][spalte-1] == 'O'))
        {
            //Feld ungueltig
            nok = 1;
            printf("Schon belegt!\n");
        }
    } while (nok);

    //Feld mit Spielfigur X markieren
    array[zeile-1][spalte-1] = 'X';
}

/*--------------------------------------------------------------------------*/
/* Funktion, die das Spielende erkennt
 * Parameter: char array[3][3], das Spielfeld
 * Rueckgabewert: 0, wenn Mensch gewonnen
 *                1, wenn Computer gewonnen
 *               -1, wenn keiner gewonnen
 */
/*--------------------------------------------------------------------------*/

int spielende(char array[3][3])
{
    int i;

    //drei in einer Reihe Mensch
    for (i = 0; i < 3; i++)
        if((array[i][0] == 'X') && (array[i][1] == 'X') &&
           (array[i][2] == 'X'))
            return 0;

    //drei in einer Reihe Computer
```

```
    for (i = 0; i < 3; i++)
        if((array[i][0] == 'O') && (array[i][1] == 'O') &&
           (array[i][2] == 'O'))
            return 1;

    //drei in einer Spalte Mensch
    for (i = 0; i < 3; i++)
        if((array[0][i] == 'X') && (array[1][i] == 'X') &&
           (array[2][i] == 'X'))
            return 0;

    //drei in einer Spalte Computer
    for (i = 0; i < 3; i++)
        if((array[0][i] == 'O') && (array[1][i] == 'O') &&
           (array[2][i] == 'O'))
            return 1;

    //drei in erster Diagonalen Mensch
    if ((array[0][0] == 'X') && (array[1][1] == 'X') &&
        (array[2][2] == 'X'))
        return  0;

    //drei in zweiter Diagonalen Mensch
    if ((array[2][0] == 'X') && (array[1][1] == 'X') &&
        (array[0][2] == 'X'))
        return 0;

    //drei in erster Diagonalen Computer
    if ((array[0][0] == 'O') && (array[1][1] == 'O') &&
        (array[2][2] == 'O'))
        return  1;

    //drei in zweiter Diagonalen Computer
    if ((array[2][0] == 'O') && (array[1][1] == 'O') &&
        (array[0][2] == 'O'))
        return 1;

    //keiner hat gewonnen
    return -1;
}

/*----------------------------------------------------------------------------*/
```

```
int main()
{
    char array[3][3];   //das Spielfeld
    int i,j;            //Zaehler
    int ende = -1;      //Ende, mit -1 (keiner gewonnen) initialisieren
    int maxzuege = 0;   //Zuege, mit 0 (kein Zug) initialisieren
    int anfang;         //wer faengt an

    //Feld mit Leerzeichen belegen
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            array[i][j] = ' ';

    //Feld ausgeben
    spielfeld(array);

    //Zufallszahlengenerator starten
    srand(time(NULL));
    //Zufallszahl holen
    anfang = rand()%2;

    //bei 0 faengt der Mensch an, sonst der Computer
    if(anfang == 0)
        printf("Mensch faengt an\n");
    else
        printf("Computer faengt an\n");

    //bis einer gewonnen hat oder neuen Zuege gemacht wurden
    do
    {
        //Mensch faengt an
        if(anfang == 0)
        {
            //Spielzug ausfuehren
            spielzugMensch(array);
            //Spielfeld ausgeben
            spielfeld(array);
            //ist schon ende?
            ende = spielende(array);
            //Zug gemacht, Anzahl der Zuege erhoehen
            maxzuege++;
```

```
        }

        //Bei Ende oder max. Anzahl von Zuegen aufhoeren
        if ((ende != -1) || (maxzuege == 9))
            break;

        //Computer ist dran
        //Spielzug ausfuehren
        spielzugComputer(array);
        //Spielfeld ausgeben
        spielfeld(array);
        //ist schon ende?
        ende = spielende(array);
        //Zug gemacht, Anzahl der Zuege erhoehen
        maxzuege ++;
        //jetzt ist der Mensch dran
        anfang = 0;

    } while((ende != 0) && (ende != 1) && (maxzuege < 9));

    //bei 0 hat der Mensch gewonnen
    if (ende == 0)
        printf("Mensch gewonnen!\n");
    //bei 1 hat der Mensch gewonnen
    else if(ende == 1)
        printf("Computer gewonnen!\n");
    //sonst untenschieden
    else
        printf("Unentschieden\n");
}
```