Übungen zur Vorlesung
GRUNDLAGEN DER INFORMATIK I
Studiengang Elektrotechnik
SS 03

AG Betriebssysteme FB3
Kirsten Berkenkötter

Ausgabe: 5.6.2003
Besprechung: 13.6.2003

Übungszettel 5

Aufgabe 1: Tic-Tac-Toe

Aufgabe ist es, ein Tic-Tac-Toe Spiel zu programmieren. Als Spielfeld wird ein Array *char spielfeld[3][3]* genommen, das zu Anfang mit lauter Leerzeichen initialisiert ist. Fuehrt der Mensch einen Zug aus, markiert er sein Feld mit *'X'*, fuehrt der Computer einen Zug aus, markiert er sein Feld mit *'O'*.

Vorgegeben ist eine Funktion zum Ausgeben des (aktuellen) Spielfeldes, der als Parameter das Spielfeld-Array übergeben werden muss:

```
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");
}
```

Die Ausgabe hat z.B. folgendes Aussehen:

```
    1   2   3
   --- --- ---
1 | X |   |   |
   --- --- ---
2 | X | O |   |
   --- --- ---
3 |   | O |   |
   --- --- ---
```

a) Schreibt eine Funktion, mit der die Spielzüge des Computers gemacht werden. Benutzt dazu Zufallszahlen, um die Zeile und Spalte zu ermitteln, in welcher der Computer seine Spielfigur setzt. Denkt daran, die Spielzüge des Computers auf Korrektheit zu überprüfen, bevor ihr sie wirklich vornehmt.

```
void spielzugComputer(char array[3][3]);
```

b) Schreibt eine Funktion, mit der die Spielzüge des Menschen gemacht werden. Fragt dazu mit *scanf* die Zeile und Spalte ab, in welche der Mensch seine Spielfigur setzen will. Macht dies solange, bis eine gültige Eingabe vorliegt.

```
void spielzugMensch(char array[3][3]);
```

c) Schreibt eine Funktion, mit der überprüft wird, ob jemand gewonnen hat, d.h. in einer Spalte, Zeile oder Diagonalen drei Spielfiguren eines Spielers stehen. Hat der Mensch gewonnen soll eine *0* zurueckgegeben werden, hat der Computer gewonnen, eine *1*. Hat keiner gewonnen, wird *-1* zurueckgegeben

```
int spielende (char array[3][3]);
```

d) Schreibt ein Hauptprogramm. Zuerst soll das Spielfeld mit Leerzeichen initialisiert und dann ausgegeben werden. Mit Hilfe von Zufallszahlen soll ermittelt werden, wer anfängt: bei einer *0* der Mensch, bei einer *1* der Computer.

Danach erfolgt das eigentliche Spiel. Der Beginner fängt an und macht seinen Spielzug, woraufhin das Spielfeld ausgegeben wird. Das gleiche gilt für den Spieler, der dann an der Reihe ist. Es soll solange gespielt werden, bis entweder ein Spieler gewonnen hat, oder die maximale Zahl der Züge gemacht wurde und ein Unentschieden resultiert.

Zu Letzt wird ausgegeben, wer gewonnen hat.

e) Im Moment spielt der Computer ausgesprochen dämlich, da er seine Züge lediglich nach Zufallsprinzip macht. Er erkennt weder, wenn sein Gegner zu gewinnen droht, noch, wenn er selbst gewinnen könnte. Um diesen Missstand zu beheben, soll eine weitere Funktion geschrieben werden, die überprüft, ob im nächsten Zug jemand gewinnen könnte.

```
int kannGewinnen(char array[3][3], int *zeile, int *spalte, char zeichen);
```

Die Funktion bekommt natürlich das Spielfeld-Array übergeben. Die Zeiger *zeile* und *spalte* dienen der Rückgabe des Ergebnisses, d.h. sie zeigen an, welches Feld gesetzt werden muss, um zu gewinnen. Der letzte Parameter *zeichen* unterscheidet, ob auf Gewinn des Computers oder des Menschen im nächsten Zug überprüft werden soll. Hier muss ein *'X'* für den Menschen und ein *'O'* für den Computer übergeben werden.

Der Rückgabewert bestimmt, ob der Test erfolgreich war oder nicht. Kann im nächsten Zug der Computer oder der Mensch gewinnen, wird eine *0* zuruckgegeben, sonst eine *-1*.

In der Funktion *spielzugComputer* wird nun zunächst abgefragt, ob im nächsten Zug entweder der Computer selbst oder der Gegner gewinnen kann. Ist dies der Fall, werden die in *zeile* und *spalte* zurückgegebenen Werte verwendet. Ansonsten werden wie gehabt Zufallswerte verwendet:

```
if((kannGewinnen(array, &zeile, &spalte, 'O') = -1) && (kannGewinnen(array, &zeile,
&spalte, 'X') == -1))

{

   do

   {

    ...
```