Übungen zur Vorlesung
GRUNDLAGEN DER INFORMATIK I
Studiengang Elektrotechnik
SS 03

AG Betriebssysteme FB3
Kirsten Berkenkötter

Ausgabe: 13.5.2003
Besprechung: 22.5.2003

Übungszettel 3

Aufgabe 1: Zahlendarstellung

Berechnet mit Hilfe von Funktionstafeln, ob die folgenden Gleichungen gültig sind oder nicht:

a) $4xy + 2(y \oplus x) + x = (x \oplus y \oplus 1) + 3x + 2y - 1$

b) $4xy + (y \oplus x) + x = (x \oplus y \oplus 1) + 3x + 2y - 1$

c) $(x \oplus y) - 3(x \oplus y \oplus 1) + 5xy = x + y + (x \oplus y)$

d) $x - y + n(x \oplus y) = n(x \oplus y) + x(x \oplus y) - y(x \oplus y)$

Aufgabe 2: Umwandlung Dezimal-Binär

Schreibt eine Funktion

void deziBinaer()

mit deren Hilfe ihr Dezimalzahlen in Binaerzahlen umwandeln könnt. Dabei soll eine Dezimalzahl vom Benutzer eingegeben (Integer-Zahl) und als Binaerzahl (Bitfeld als Character-String) ausgegeben werden. Die Umwandlung erfolgt wie es in der Übung gezeigt wurde.

Um das Bitfeld als String augeben zu können, müssen die binäre Null als letztes Zeichen eingefügt (markiert das Ende eines Strings) und die Zeichenkette 'umgedreht' werden. Dazu könnt ihr die Funktion *binToString(char feld[], int laenge)* verwenden, die im folgenden vorgegeben ist:

```
/* Funktion zur Umwandlung eines Bitfeldes in einen String
 * Parameter: char feld[], Bitfeld
 *            int laenge, Laenge des Bitfeldes
 */
void binToString(char feld[], int laenge)
{
    int i;
    //Hilfsarray
    char hilf[21];

    //binaere 0 einfuegen, markiert Stringende
    hilf[laenge] = '\0';
```

```
    //Bits in umgekehrter Reihenfolge in das Hilfsarray schreiben
    for (i = 0; i < laenge; i++)
        hilf[laenge - i - 1] = feld[i];

    //Bits als String in das urspruengliche Feld schreiben
    for (i = 0; i < laenge + 1; i++)
        feld[i] = hilf[i];
}
```

Aufgabe 3: Umwandlung Binär-Dezimal

Schreibt eine Funktion

void binaerDezi()

mit deren Hilfe ihr Binärzahlen in Dezimalzahlen umwandeln könnt. Dabei soll eine Binärzahl vom Benutzer eingegeben (Character-Feld) und als Dezimalzahl (Integer) ausgegeben werden.

Um die Zweierpotenzen zu bilden, könnt ihr die Funktion *int zweierpotenz(int exponent)* verwenden, die im folgenden dargestellt ist:

```
/* Funktion zur Berechnung von 2^n
 * Parameter: int exponent, (n)
 * Rueckgabewert: 2^n
 */
int zweierpotenz(int exponent)
{
    //2^0 ist 1, deshalb Rueckgabewert mit 1 vorbelegen
    int potenz = 1;
    int i;

    //2^n berechnen
    for (i = 1; i <= exponent; i++)
        potenz *= 2;

    //2^n zurueckgeben
    return potenz;
}
```