

## Lösungen zu Übungsblatt 2

### Aufgabe 1:

Suche alle Fehler in der folgenden Funktion (kleiner Anhaltspunkt: Es sind 5 Fehler).

(Fehler sind **rot** hervorgehoben)

```
int hoechste_zahl(int zahlen[], int n)
{
    int max = -9999;

    for(int i=0; i < n; i++) // an dieser Stelle muss es entweder i++ oder i = i+1 heißen
    {
        if(zahlen[i] > max)
            max = zahlen[i];
    }

    cout << "Hoechste Zahl: " << max << endl;

    return max;
}
```

### Aufgabe 2:

Schreibe ein Programm, das den Benutzer nach Matrixwerten fragt und die eingegebenen Werte anschließend in bekannter Matrixform darstellt:

Hier der Quelltext:

```
#include <iostream>
#include <conio.h>

using namespace std;

const int N = 3;

void eingeben(int matrix[N][N])
{
    for(int i=0; i<N; i++)
    {
        for(int j=0; j<N; j++)
        {
            cout << "Wert " << i+1 << ", " << j+1 << ": ";
            cin >> matrix[i][j];
        }
    }
}
```

```

void ausgeben(int matrix[N][N])
{
    for(int i=0; i<N; i++)
    {
        for(int j=0; j<N; j++)
        {
            cout << "(" << matrix[i][j] << ") ";
        }
        cout << endl;
    }
}

main()
{
    int matrix[N][N];
    cout << endl;

    cout << "Eingabe der Matrixwerte: " << endl;
    eingeben(matrix);
    cout << endl << "Ausgabe der eingegebenen Werte in Matrixform: " << endl << endl;

    ausgeben(matrix);
    getch();
}

```

*Zur Erklärung:*

*Bei einer nxn-Matrix (sprich: n-kreuz-n-Matrix) muss vorher angegeben werden, wie groß n ist. Das muss hierbei "global" festgelegt werden. Was bedeutet "global"? Das bedeutet, dass es nicht in einer Funktion deklariert wird, sondern außerhalb der Funktion. Eine in einer Funktion definierte Variable ist auch nur innerhalb der Funktion gültig. Eine global definierte Variable ist im gesamten Programm (und somit in JEDER Funktion) gültig.*

*Da es sich beim Wert für n um eine Konstante handelt, wird n groß geschrieben (ist reine Formsache, man hat sich halt mal drauf geeinigt, dass Konstanten groß geschrieben werden).*

### **Aufgabe 3:**

a)  
Schreibe die Funktion ***int verdopple(int zahl)***, die eine übergebene Zahl verdoppelt und wieder zurück gibt.

```

int verdopple(int zahl)
{
    return 2*zahl;
}

```

***alternativ geht auch:***

```

int verdopple(int zahl)
{
    int erg = 2 * zahl;
    return erg;
}

```

b)

Schreibe nun die Funktion **void zahlenarray(int zahlen[], int n)**, die alle n Elemente des übergebenen Arrays (also des Arrays zahlen[]) durchgeht und zum verdoppeln der Werte die Funktion aus Aufgabe a) benutzt.

```
void zahlenarray(int zahlen[], int n)
{
    for(int i=0; i<n; i++)
    {
        zahlen[i] = verdopple(zahlen[i]);
    }
}
```

**Vorschlag für eine vernünftige main-Funktion:**

```
main()
{
    cout << "Anzahl der Arrayelemente: ";
    int n;
    cin >> n;

    int array[n];

    cout << "Werteingabe ins Array: " << endl;
    for(int i=0; i<n; i++)
    {
        cout << "Wert " << i+1 << ": ";
        cin >> array[i];
    }

    zahlenarray(array, n);

    cout << endl << "Werte wurden nun verdoppelt. Ausgabe der verdoppelten Werte: " << endl;
    getch(); // Wartet auf ein Zeichen, bevor es weiter geht

    for(int i=0; i<n; i++)
    {
        cout << "[" << array[i] << "] ";
    }

    getch();
}
```