3. Unterschiede zwischen Funktionen mit Rückgabewerten und ohne

Eigenschaft	**mit**	**ohne**
Ergebnis über	Funktionsnamen	&-Argumente
Anzahl der Ergebnisse	eins	beliebig viele
return-Anweisung	unbedingt nötig	verboten
Aufruf	in Ausdruck	als Anweisung

Anmerkung: Jede Funktion mit Rückgabewerten kann in eine Funktion ohne Rückgabewerte umformuliert werden. Die Umkehrung dieses Satzes ist i. a. nicht richtig.

Aufgabe 6:
Schreiben Sie die Funktion ***ggt*** aus Aufgabe 3 zur Berechnung des größten gemeinsamen Teilers in eine Funktion ohne Rückgabewerte um und lösen Sie Aufgabe 3 mit dieser Funktion.

Aufgabe 7:
Schreiben Sie das Programm zur Berechnung perfekter Zahlen (Aufgabe 6, S. 9) so um, dass die Ermittlung einer perfekten Zahl *n* durch die Funktion ***BerechnePerfekt*** (ohne Rückgabewerte) mit formalen Parametern *n* (Typ *int*) und *b* (Typ *bool*) erfolgt. Der Funktionsaufruf soll dabei wie folgt aussehen: ***BerechnePerfekt(zahl, ist)***.

4. Lokale und globale Variablen

Lokale Variablen einer Funktion werden innerhalb des Funktionsrumpfes deklariert und existieren nur innerhalb dieser Funktion. Beim Verlassen der Funktion stehen sie nicht mehr zur Verfügung.

Globale Variablen werden außerhalb aller Funktionen deklariert (wegen der Übersichtlichkeit) möglichst gleich am Anfang und sind für jede Funktion – einschließlich *main* – verfügbar.

Zu jeder Variablen gehört also ein **Gültigkeitsbereich**, der bestimmt, wie lange eine Variable in einem Programm zugänglich ist. Der Gültigkeitsbereich einer in einem Block deklarierten Variablen beschränkt sich auf diesen Block.

Aufgabe 8:
Was wird durch folgendes Programm auf dem Bildschirm ausgegeben? Versehen Sie das Programm mit passenden Kommentaren. Was passiert, wenn man vor *return 0;* den Befehl *cout << y << endl;* einfügt?

```
//gueltig.cpp
#include <iostream.h>
int main()
{//1. Block Beginn
        int x = 7;
        {//2. Block Beginn
                int y = 10;
                cout << x << endl << y << endl;
                int x = 14;
                cout << x << endl << y << endl;
        }//2. Block Ende
        cout << x << endl;
        return 0;
}//1. Block Ende
```

Folgendes Programm in **schlechtem Programmierstil (!!!)** soll der Demonstration von **Seiteneffekten** dienen, die bei der Verwendung globaler Variablen in Funktionen auftreten können.

```
#include <iostream.h>
int a, b, c;                                    //3 globale Variablen
int GlobalAendern(int inX)
{
        a = a +1;
        return a * inX;
}
int main()
{
        a = 10;
        b = 3;
        c = GlobalAendern(b) + GlobalAendern(b);        //diese Zeile ersetzen
        cout << c << endl;
        return 0;
}
```

Aufgabe 9:
Machen Sie einen Variablenbelegungsplan für obiges Programm. Ersetzen Sie die angegebene Zeile durch ***c = 2 * GlobalAendern(b);***. Wie sind nun die Variablen nach Abarbeitung des Programms belegt?

5. Header-Dateien

Häufig verwendete Funktionen (globale Variablen, Klassen) können in separaten Header-Dateien (mit Dateiendung *h*) abgespeichert werden und über *include* in cpp-Programme wie folgt eingebunden werden (im Beispiel befindet sich *mein.h* in demselben Ordner wie *mein.cpp*).

```
//mein.cpp
#include <iostream.h>
#include "mein.h"

int main()
{       Funktionsrumpf, benutzt Definitionen aus mein.h     }
```

Aufgabe 10:
a) Kopieren Sie das Programm *ggt.cpp* und bezeichnen Sie die kopierte Version mit *ggt3.cpp*.
b) Erstellen Sie die header-Datei *mein.h*.
c) Schneiden Sie die Funktion zur Berechnung des größten gemeinsamen Teilers aus *ggt3.cpp* aus und fügen Sie sie in *mein.h* ein.
d) Binden Sie *mein.h* in *ggt3.cpp* ein und testen Sie das Programm.

Aufgabe 11:
Ergänzen Sie *mein.h* durch die Funktion *tausch* (aus Aufgabe 1, S. 10).

Aufgabe 12:
Definieren Sie in *main.h* die Funktion *fakul*, die $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ iterativ berechnet, und schreiben Sie ein Programm, das nach Eingabe einer beliebigen natürlichen Zahl *n* durch Aufruf der Funktion *fakul* die Zahl *n*! ermittelt und auf dem Bildschirm ausgibt.