Funktionen (Abstraktion)

- Funktionen ermöglichen es, Programme in kleinere Einheiten zu zerlegen, um die Übersichtlichkeit und Wartbarkeit zu erhöhen.
- Funktionen besitzen i. a. **Eingabewerte** und **Rückgabewerte**.
- Das Hauptprogramm selbst ist auch eine Funktion (mit dem Namen ***main***).
- Das Schlüsselwort ***void*** legt fest, dass die Funktion keine Rückgabewerte hat.
- Soll eine Funktion Werte zurückgeben, dann geschieht dies über die ***return***-Anweisung.
- Es gibt **Standardfunktionen**, wie z. B. die mathematischen Funktionen *cos*, *sin*, *exp*, *pow* usw. aus <*math.h*>.
- Man kann sich aber auch Funktionen selbst definieren.

1. Funktionen ohne Rückgabewerte

Beispiel:

```
//Funktion zum Vertauschen der Werte zweier Variablen
void tausch(int& a, int& b)   //Funktionskopf mit zwei formalen Referenzparametern
{
  int hilf;                   //Vereinbarung einer lokalen Variablen
  hilf = a;                   //Anweisungsteil der Funktion
  a = b;
  b = hilf;
}
```

Aufgabe 1:
Erstellen Sie ein Programm, das nach Eingabe von Werten für die Integer-Variablen *x* und *y* diese mit Hilfe der obigen Funktion tauscht. Hinweis: Der **Funktionsaufruf** erfolgt durch die Anweisung ***tausch*(*x*, *y*);**

Formale Parameter: stehen im Funktionskopf. Mit ihrer Hilfe wird festgelegt, wie die aktuellen Parameter aus der übergeordneten Funktion zu verarbeiten sind.
Aktuelle Parameter: treten beim Aufruf der Funktion auf.

Bei den formalen Parametern unterscheidet man **Referenzparameter** und **Wertparameter.**

Eigenschaft	**Wertparameter**	**Referenzparameter**
Syntax	**ohne** &	**mit** &
Aktuelles Argument	**darf** Ausdruck sein	**muss** Variable sein
Zuweisung an das formale Argument	lässt das aktuelle Argument **unverändert**	**verändert** das aktuelle Argument
Verwendung des Arguments	**Übergabe** von Werten aus dem aufrufenden Programm	Übergabe und **Rückgabe** von Werten

Aufgabe 2: Testen Sie folgendes Programm:

```
//untersch.cpp
//Unterschied zwischen Referenz- und Wertparametern
#include <iostream.h>
void aendertnichts(int a);      //Prototyp der Funktion aendertnichts
void aendertwas(int& a);        //Prototyp der Funktion aendertwas

int main()
{
        int x;
        x = 5;
        cout << x << endl;
        aendertwas(x);
        cout << x << endl;
        aendertnichts(x);
        cout << x << endl;
        cout << endl;
        return 0;
}
//Funktionsdefinitionen
void aendertnichts(int a)
{
        a = 2 * a;
        cout << a << endl;
}
void aendertwas(int& a)
{
        a = 2 * a;
        cout << a << endl;
}
```

2. Funktionen mit Rückgabewerten

Beispiel für eine Funktionsdefinition mit Rückgabewerten:

```
int  ggt(int m, int n)          //void wird durch den Ergebnistyp int ersetzt
{
        int r;
        do
        {
                r = m % n;
                m = n;
                n = r;
        }
        while (r != 0);
        return m;               //return ermöglicht Rückgabe des Wertes an die
                                //aufrufende Stelle und beendet die Funktion
}
```

Aufgabe 3:
Erstellen Sie ein Programm, das nach Eingabe zweier natürlicher Zahlen *a* und *b* mit der Funktion ***ggt*** den größten gemeinsamen Teiler dieser beiden Zahlen ermittelt und ausgibt.

Aufgabe 4:
Welche Werte werden durch das folgende Programm ausgegeben? Geben Sie eine Speicherbelegungstabelle an.

```
//fktsumme.cpp
#include <iostream.h>
int summe (int zahl);
int main()
{
        int k;
        for (k = 16; k <= 18; k++)
                cout << k << "  " << summe(k) << endl;
        return 0;
}
int summe (int z)
{
        int a, b, s;
        s = z + 1;
        a = 2;
        b = z / 2;
        while (a < b)
        {
                if (a * b == z) s = s + a + b;
                a = a + 1;
                b = z / a;
        }
        if (z == a * a) s = s + a;
        return s;
}
```

Aufgabe 5:
Gegeben sei eine quadratische Gleichung $ax^2 + bx + c = 0$.
Für reelle Zahlen a, b, c (mit $a \neq 0$) kann man die Lösungen, falls sie existieren, nach der Formel

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

berechnen. Eine reelle Lösung existiert, wenn der Ausdruck unter der Wurzel nicht negativ ist.

Schreiben Sie ein Programm, das die Funktion *LoesungBestimmen* enthält. Die Funktion soll die sechs formalen Parameter *a*, *b*, *c*, *loes*1, *loes*2 (alle vom Typ *double*) und *gibtLoes* (Typ *bool*) besitzen für die Übernahme der Koeffizienten und die Berechnung der Lösungen, falls diese existieren. Kommentieren Sie das Programm ausführlich. Bezeichnen Sie die aktuellen Parameter anders als die formalen Parameter.
Die Ausgabe auf dem Bildschirm sollte etwa wie folgt aussehen:

```
Loesungen einer quadratischen Gleichung
Geben Sie die Koeffizienten ein:

a:   1.0
b:   3.0
c:   2.0

Loesungen:      -1.00 und -2.00
```