Die Funktionsdefinition

Eine Funktion ist ein Unterprogramm. Die Funktionsdefinition enthält den kompletten Code der Funktion. Sie muss außerhalb der main-Funktion stehen, gewöhnlich darunter. Später werden wir sie in einer eigenen Datei unterbringen.

Im **Funktionskopf** – der ersten Zeile – steht der Name der Funktion. Falls die Funktion einen Wert mit dem Befehl `return` zurückgibt, steht vor dem Namen der Rückgabetyp. Wenn kein Wert zurückgegeben wird, schreibt man stattdessen `void`.

Einer Funktion können Werte übergeben werden. Diese sogenannten **Parameter** werden mit Typ und Name in Klammern nach dem Funktionsnamen angegeben. Falls kein Wert übergeben wird, lässt man das Klammerpaar leer.

Im **Funktionskörper** stehen innerhalb von geschweiften Klammern die Anweisungen, die in der Funktion ausgeführt werden sollen.

Beispiel für eine Funktionsdefinition:

```
int addiere(int a, int b)               // Funktionskopf mit Parametern
{                                       // ab hier Funktionskörper
          int ergebnis = a+b;
          return ergebnis;              // return (entfällt bei void)
}
```

Den Funktionskopf können Sie mit aktuellen Compilern (seit C++11) auch anders schreiben:

```
auto addiere(int a, int b) -> int
```

Die Funktionsdeklaration

Der Compiler braucht ein Muster für den Funktionsaufruf, einen sogenannten Prototyp. Er besteht aus einer Kopie des Funktionskopfes gefolgt von einem Strichpunkt. Die Parameternamen darf man dabei weglassen, aber nicht die Datentypen.

Diese Funktionsdeklaration schreibt man in der Regel vor die main-Funktion.

Beispiel:

```
int addiere(int a, int b);              // Funktionsdeklaration (Prototyp)

int main()                              // Hauptprogramm
{
          // ... viele Zeilen ...
          return 0;
}

int addiere(int a, int b)               // Funktionsdefinition
{
          int ergebnis = a + b;
          return ergebnis;
}
```

Kurze Funktionen können Sie auch komplett vor das Hauptprogramm schreiben. Dann fällt die Deklaration weg.

Der Funktionsaufruf

Funktionen werden im Hauptprogramm oder in anderen Funktionen mit ihrem Namen und mit Werten für die Parameter aufgerufen.

Beispiel:

```
int addiere(int a, int b);

int main()
{
          int x = 39, y = 3;
          int summe;

          summe = addiere(x, y);        // Funktionsaufruf
          cout << summe << endl;

          return 0;
}

int addiere(int a, int b)
{
          int ergebnis = a+b;
          return ergebnis;
}
```

Im Beispiel wird der Wert von `x` aus dem Hauptprogramm in den Parameter `a` der Funktion kopiert. Der Wert von `y` wird in `b` kopiert.

Man kann eine Funktion auch als Anweisung verwenden und den Rückgabewert ignorieren.

Variablen in der Funktionsdefinition

In jeder Funktion darf man Variablen definieren, so wie `ergebnis` im Beispiel. Diese sind völlig unabhängig von den Variablen anderer Programmteile, auch wenn sie den gleichen Namen haben.

Unterprogramme in anderen Sprachen

Viele andere Hochsprachen unterscheiden zwei Formen von Unterprogrammen, nämlich Funktionen mit Rückgabewert und Prozeduren ohne Rückgabewert. In C++ nennt man beides Funktion.

In objektorientierten Sprachen kommen noch sogenannte Methoden dazu. Damit werden wir uns später beschäftigen.