## Funktionen

| Funktion | **Selbstständiger Codeabschnitt** |
|---|---|

Wichtige Befehle:

**Definition:** `int my_fun (bool arg1, float arg2) {...}`
**Rückgabe:** `return my_val;`
**Aufruf:** `my_fun(true, 3.75f)`

Der Rückgabewert wird immer zum Rückgabetyp konvertiert.

Jede Funktion, die nicht den Rückgabetyp `void` hat, muss ein `return` haben.

```
int bin_digits (int n) {
    assert(n >= 0); // do not accept negative numbers
    if (n == 0)
        return 1; // stops function and returns 1
    int count = 0;
    do {
        n /= 2;
        ++count;
    } while (n > 0);
    return count;
}

int main () {
    std::cout << bin_digits(3) << "\n"; // Output: 2
    std::cout << bin_digits(8) << "\n"; // Output: 4
    return 0;
}
```

| `// PRE: ...`<br>`// POST: ...` | **Funktionsbeschreibung** |
|---|---|

PRE-/POST-Conditions gehören vor **jede** Funktionsdefinition ausser der `main`-Funktion. (In diesen Programmier-Befehlszusammenfassungen werden sie aber manchmal aus Platzgründen weggelassen.)

Man kann beispielsweise `assert` verwenden, um das Programm abzubrechen, falls die Funktion doch mal mit Argumenten aufgerufen wird, welche die PRE-Condition verletzen.

( ... )

( ... )

```
// POST: return value is a^4
int power_4 (int a) {
    return a*a*a*a;
}

// PRE: width >= 0 and height >= 0
// POST: returns the rectangle area given by width and height
double area (double width, double height) {
    assert(width >= 0 && height >= 0);
    return width * height;
}
```

## Datentypen

| `void` | Datentyp für **Funktion ohne Rückgabe**. |
|---|---|
| `void`-Funktionen haben keinen Rückgabewert, aber **sinnvollerweise einen Effekt** (z.B. Textausgabe im Beispiel unten). | |

```
void print_account (double assets, double interest) {
    std::cout << "Your assets:   " << assets << "\n"
              << "Your interest: " << interest << "\n";
}
```