# Grundlagen des Programmierens

Programmieren ist das Entwerfen eines Algorithmus und das Implementieren (Codieren) desselben als Programm zur Lösung eines gegebenen Problems mit dem Computer.

Ein Algorithmus ist eine Vorschrift zur Durchführung von Aktionen an Objekten.
Beispiele für "Alltags-Algorithmen":
- Kochrezept: Man nehme ..., gebe es in einen Topf, erhitze solange bis ... usw.
- Bedienungsanleitung Telefon: Hörer abheben, wählen, falls Besetztzeichen ertönt auflegen, anderenfalls warten solange bis Gesprächspartner abhebt, dann sprechen ...

Beim Programmieren sind die Objekte (im Wesentlichen) Daten, die Aktionen sind Operationen wie Speichern, Kopieren, Vergleichen, Addieren, Subtrahieren, ...
Programmieren beginnt mit der Festlegung einer Methode zur Lösung eines gegebenen Problems. Diese Methode (Algorithmus) sollte programmiersprachenunabhängig beschrieben werden. Der zweite Schritt ist die Umsetzung eines Algorithmus in eine konkrete Programmiersprache (Implementieren, Codieren).

## Programmentwicklung:

1. Problemanalyse
2. Datenanalyse: Eingabedaten, Ausgabedaten, Beziehungen/Operationen zwischen Ein- und Ausgabedaten
3. Entwicklung des Algorithmus
4. Formulierung des Programms (Struktogramm)
5. Eingabe des Programms und Programmtest (Syntaxtest, Logiktest)
6. Dokumentation (vor und im Programm mit Kommentaren)

## Die sieben Elemente der Programmierung

Eingabe — Lesen von Daten von einer Festplatte, Diskette, Tastatur oder über die Schnittstelle.

Ausgabe — Schreiben von Daten auf Bildschirm, Festplatte, Diskette, Drucker oder Schnittstelle.

Datentypen — Konstanten, Variablen und Strukturen, die numerische Werte (Integer- und Realzahlen), Text (Zeichen und Strings) oder Adressen enthalten.

Operationen — Zuweisungen von Werten an Variablen, Verknüpfungen von Werten (Addition, Division, etc.), sowie Vergleiche (gleich, größer, ungleich,...).

Bedingte Ausführung: Die Ausführung von Anweisungen nach einer Operation (z.B. Vergleich). Ist die Operation wahr, so wird die Anweisung ausgeführt, ansonsten übersprungen.

Wiederholte Ausführung (Schleifen): Wiederholung von Anweisungen, entweder fest vorgegeben oder abhängig von einer Bedingung .

Unterprogramme (Prozeduren, Funktionen): Gruppen von Anweisungen, die genau wie ein vollständiges Programm aufgebaut sind und die von mehreren Stellen des Programms aufgerufen werden können, wobei eine Übergabe der Daten stattfindet.

## Programmstrukturen ( Programmentwurf mit Struktogrammen )

Elementare Anweisung

| Anweisung |
|---|

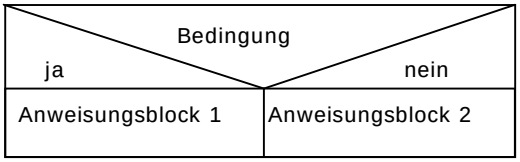
Einfache sequentielle Aufeinanderfolge
elementarer Anweisungen,
z.B. Wertzuweisungen, Berechnungen, ...

| Anweisung 1 |
|---|
| Anweisung 2 |
| Anweisung .... |

Steueranweisungen:

Bedingte Ausführung, Alternative

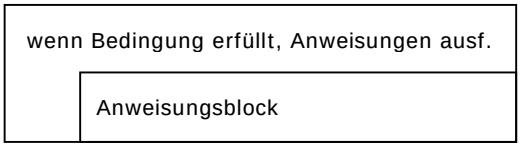
Abhängig von der Bedingung wird entweder nur Anweisungsblock 1 oder nur Anweisungsblock 2 ausgeführt




Wiederholungs-Anweisungen, Schleifen

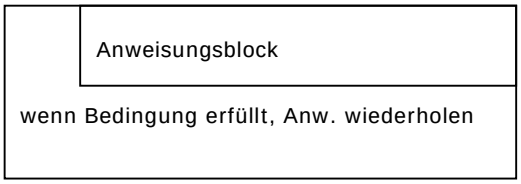
Vor der Ausführung des Anweisungsblocks wird geprüft, ob die Bedingung zutrifft, erst dann wird der Anweisungsblock ausgeführt.

Schleifen dieses Typs können null- bis unendlich mal durchlaufen werden.

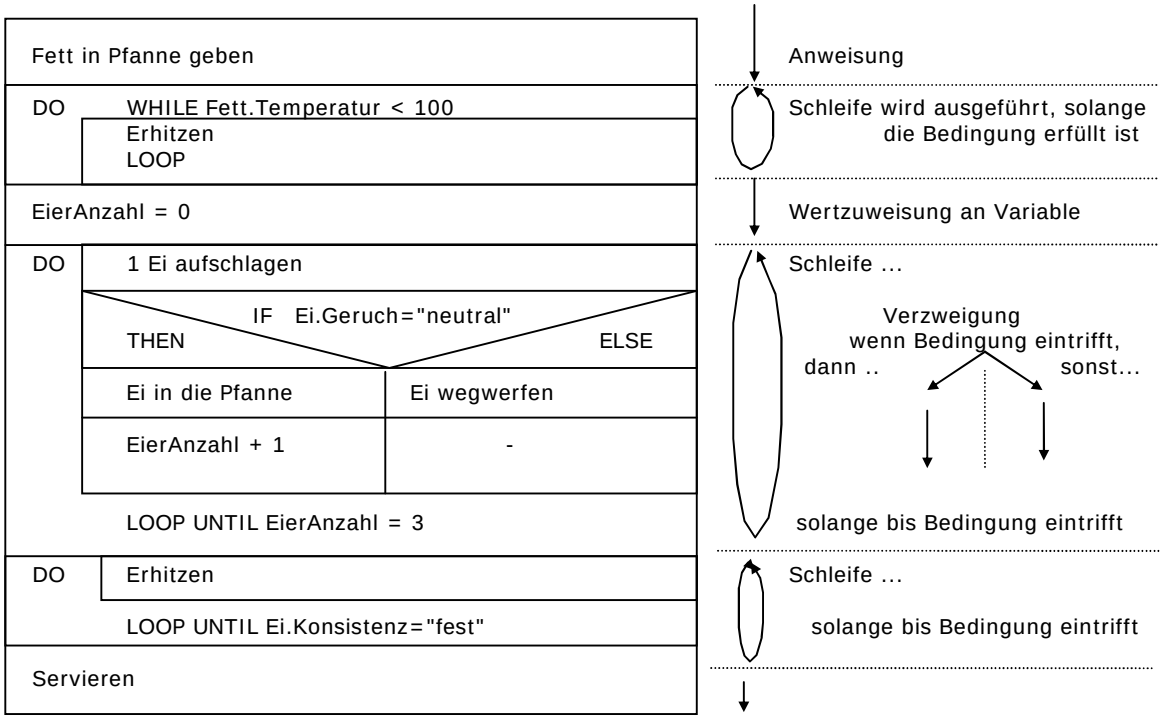



Zuerst wird der Anweisungsblock ausgeführt, dann die Bedingung überprüft, und gegebenenfalls der Anweisungsblock wiederholt.

Schleifen dieses Typs werden mindestens einmal durchlaufen.




Beispiel eines „Alltags-Algorithmus“ (Herstellen einer Eierspeise von 3 Eiern):



Eine Variable ist ein Speicherplatz, dem Werte zugewiesen werden können. Im folgenden Beispiel wird der Variablen a wird der Wert 5 zugewiesen, der Variablen b der Wert 3. Das Gleichheitszeichen ist hier als Wertzuweisung zu verstehen ( a ← 5; in andern Programmiersprachen z.B. a:=5 ).

```
a=5                Wertzuweisung an Variable a ← 5
b=3                        -"-               b ← 3

PRINT a            Ausgabe des Inhalts der Variablen a  am Bildschirm      5
PRINT b                                              b                     3

c = a + b          Wertzuweisung an Variable c ← 5 + 3
PRINT a, b, c                                                    5     3     8
```

Die Programmzeile c=a+b ist nicht als Gleichung im mathematischen Sinn zu interpretieren, sondern folgendermaßen: Die aktuellen Werte der Variablen a und b werden addiert, und dieser Wert wird der Variablen c zugewiesen ( c ← 5 + 3, in manchen Programmiersprachen c := 5 + 3 ) .
Siehe dazu auch folgende Beispiele:

Bei einer Wertzuweisung wird *zuerst* der *rechte* Ausdruck ausgewertet, und dann der Variablen *links* vom Gleichheitszeichen zugewiesen.
So ist auch der scheinbar widersprüchliche Ausdruck i=i+1 zu verstehen.

*Variable* ← = *Ausdruck*

```
i=17          Wertzuweisung an Variable  i ← 17
PRINT i       Ausgabe des Inhalts der Variablen  i am Bildschirm: 17
i = i + 1     Arithmetischer Ausdruck  i+1 wird berechnet, und das Ergebnis der Variablen i zugewiesen:
              i ← 17+1
PRINT i       Ausgabe des Inhalts der Variablen  i  am Bildschirm: 18
```

Sie finden im Folgenden ein Beispiel eines Programmablaufs. Neben jeder Zeile BASIC-Code ist der *aktuelle Inhalt* der verwendeten Variablen zu sehen. Ein solche Tabelle ist ein Hilfsmittel, um sich Schritt für Schritt anzusehen, was in einem Programm vor sich geht, und daher sehr gut geeignet, um logische Fehler in einem Programm aufzuspüren, oder um die Funktionsweise eines Programms besser zu verstehen.

```
PRINT "i", "Faktorielle(i)"
i=1
f=1
DO WHILE i<=3
   f=f*i
   PRINT i, f
   i=i+1
   LOOP
PRINT "Ende"
```

```
i             Faktorielle(i)
 1             1
 2             2
 3             6
Ende
```

| BASIC-Anweisung | Inhalt der Variablen **i** nach Ausführung der Anweisung | Inhalt der Variablen **f** nach Ausführung der Anweisung | Kommentar |
|---|---|---|---|
| print "i Faktorielle" | - | - | Ausgabe einer Stringkonstante am Bildschirm |
| i=1 | 1 | - | Wertzuweisung an Variable i ← 1 |
| f=1 | 1 | 1 | Wertzuweisung an Variable f ← 1 |
| do while i<=3 | 1 | 1 | Schleifenbedingung erfüllt, da i<=3 |
| f=f*i | 1 | 1 | Aktueller Wert von f ist 1 und von i ist 1, 1 mal 1 ergibt 1, wird der Variablen f zugewiesen: f ← 1 mal 1 |
| print i, f | 1 | 1 | Ausgabe: 1 1 |
| i=i+1 | 2 | 1 | Aktueller Wert von i ist 1, dazu wird 1 addiert, ergibt 2, wird der Variablen i zugewiesen: i ← 1 + 1 |
| loop → do while i<=3 | 2 | 1 | Schleife, solange die Bedingung i<=3 erfüllt ist. |
| f=f*i | 2 | 2 | Wertzuweisung: f ← 1 mal 2 |
| print i, f | 2 | 2 | Ausgabe: 2 2 |
| i=i+1 | 3 | 2 | Wertzuweisung: i ← 2 + 1 |
| loop → do while i<=3 | 3 | 2 | Schleife |
| f=f*i | 3 | 6 | Wertzuweisung: f ← 2 mal 3 |
| print i, f | 3 | 6 | Ausgabe: 3 6 |
| i=i+1 | 4 | 6 | Wertzuweisung: i ←3 + 1 |
| loop ↓ | 4 | 6 | Schleife wird verlassen, da i>3 und die WHILE-Bedingung somit nicht mehr erfüllt ist. |
| print "Ende" | 4 | 6 | Ausgabe eines Strings am Bildschirm |

Variablen und Konstanten haben einen bestimmten Datentyp (siehe auch Seite 5 !!). Jede Programmiersprache kennt etwas unterschiedliche Datentypen, im Prinzip gibt es aber (fast) immer den Datentyp Zahl (meist unterschieden in Ganzzahl und Fließkomma) und den Datentyp Text.

```
a = 2.5          Dezimal*punkt*, nicht Komma!
b = 4.5
c = "27"         Der Variablen c wird der Text 27 zugewiesen und nicht die Zahl 27 !!!!
PRINT a + b      Arithmetischer Ausdruck      7
PRINT "a + b"    Zeichenfolge, String         a + b
```

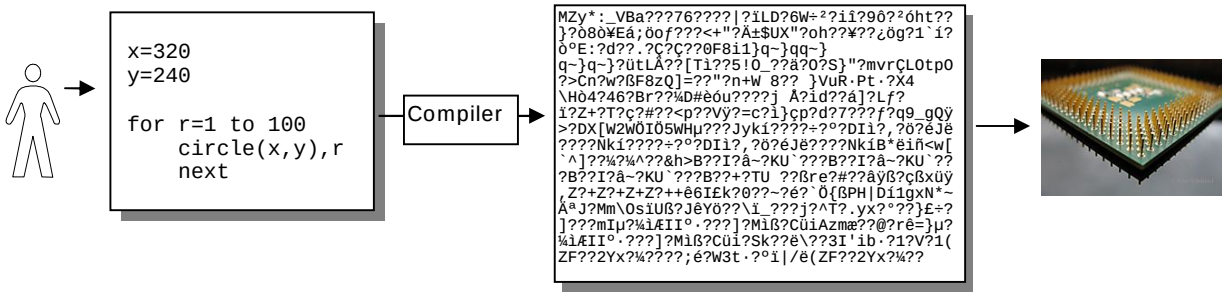
Alles, was zwischen den Hochkommas ".." steht, wird nicht ausgewertet, sondern so wie es ist als Text betrachtet. Hingegen wird a+b als arithmetischer Ausdruck betrachtet und ausgewertet.

## Maschinensprache / Höhere Programmiersprachen

Der Prozessor ist „nur“ ein elektronischer Bauteil, der einfachste Operationen (Maschinenbefehle) ausführen kann. Eine für den Menschen einfach zu formulierende Aufgabe wie z.B. das Zeichnen eines Kreises muss dem Prozessor erst langwierig mit Hilfe tausender elementarer Maschinenbefehle „beigebracht“ werden.
Da das direkte Programmieren in Maschinensprache sehr mühsam und aufwendig ist, wurden sogenannte Höhere Programmiersprachen entwickelt, die es erlauben, Aufgaben prägnanter zu formulieren. Ein in einer höheren Programmiersprache geschriebenes Programm (Quellprogramm, Source-Code) ist für den Prozessor nicht direkt verständlich, sondern muss erst in eine Fülle von einfachsten, elementaren Maschinenbefehlen übersetzt werden:

Ein Compiler ist ein Programm, das in höheren Programmiersprachen geschriebene Programme in Maschinenprogramme (.EXE oder .COM-Dateien) übersetzt.

Ein Interpreter (wie z.B. VBA) erzeugt keine .EXE-Dateien, sondern interpretiert Zeile für Zeile des Programms und setzt sie in entsprechende Aktionen um.




Mensch → Quellprogramm  Compiler → Maschinenprogramm (.EXE ) → Prozessor

Beispiele für Programmiersprachen ( → siehe Vorlesung! )

- Visual Basic
- C, C++ (sprich: "Ce plusplus"), C# (sprich: "Ce Sharp")
- Java
- Perl
- PHP
- FORTRAN
- …