Lernen zu programmieren heißt, die Syntax einer Programmiersprache - d.h. die Struktur und Form gültiger Programme - und ihre Semantik - d.h. ihre Wirkung zu verstehen, um mit diesem Wissen korrekte, verständliche und effiziente Programme entwickeln zu können.

(Lutz Wegner, Uni Kassel, 1998)

Wirkprinzipien des Computers sind Programmierbarkeit der Maschine und Algorithmierbarkeit des Problems[1].

Nach DIN 44300 ist ein Programm eine „**nach den Regeln der verwendeten Programmiersprache festgelegte syntaktische Einheit aus Anweisungen und Vereinbarungen, welche zur Lösung einer Aufgabe notwendige Elemente umfasst**“[2]. Eng mit dem Programm verknüpft sind Begriffe wie: Algorithmus, Daten, Datei, Variablen Datentypen, Resultate, Software.

Prinzipiell löst man beim Programmieren ein Problem mithilfe einer Programmiersprache. Ein Programm ist somit die Umsetzung eines Algorithmus und der zugehörigen Datenbereiche in eine Programmiersprache. Während Algorithmen allgemein beschrieben werden können, sind Programme wesentlich konkreter.

Dabei stellt sich die Frage, welche Probleme mithilfe eines Computers lösbar sind und welche nicht. Dazu gibt es umfangreiche theoretische Untersuchungen - viele davon werden Sie in den "Grundlegenden Konzepten der Informatik" diskutieren. Ein Algorithmus ist eine eindeutige Handlungsvorschrift, die aus endlich vielen einzelnen Schritten besteht und ein Problem löst.

Wie jede lebende Sprache, folgt auch eine Programmiersprache bestimmten **Regeln**. Eine Sprache besitzt: Lexik, Syntax, Semantik und Pragmatik[3].

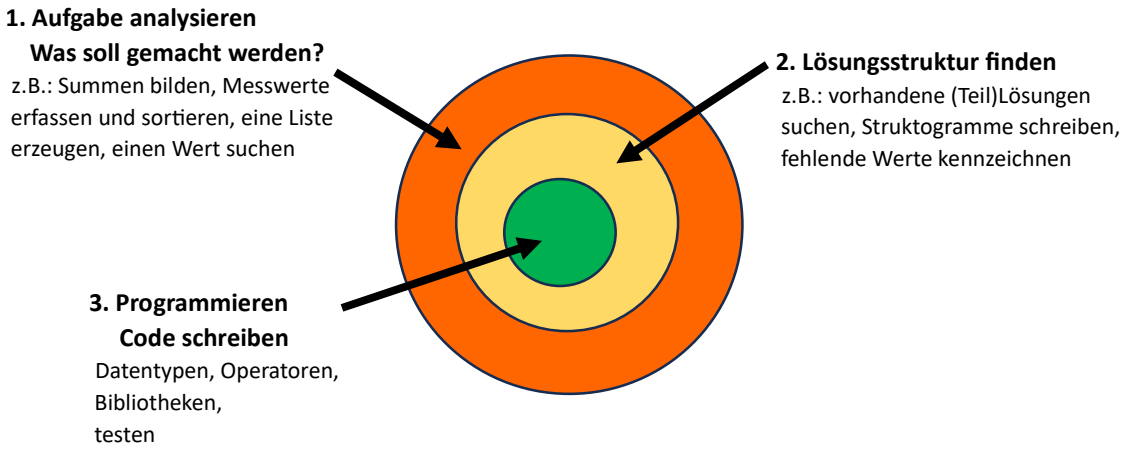
- **Lexik**: bestimmt den Aufbau der Grundelemente (Alphabet = verfügbare Zeichen),
- **Syntax**: Form und Struktur der Sprache, zulässige Ausdrucke (Schlüsselwörter), Einrückungen, Trenn- und Satzzeichen, Groß- und Kleinschreibung,
- **Semantik**: ordnet den Ausdrucken eine Bedeutung zu, Was soll das Programm ausführen?
- **Pragmatik**: Beziehung der Sprache zum subjektiven Nutzer (Erlern und Lesbarkeit des Codes

[1] Vgl. Horn 2001, S. 22

[2] DIN 44300 abgelöst durch ISO/IEC 2382:2015 (Information technology – Vocabulary)

[3] Vgl. Horn 2001, S. 275 ff.

Die Drei Ebenen der Programmerstellung



Der typische Ablauf bei der Programmerstellung[4] nach dem Wasserfallmodell

1. **Problem verstehen:** Was soll das Programm leisten?
   Lastenheft = Was will der Anwender?
   Pflichtenheft = Wie erfolgt die Umsetzung durch die Programmentwicklung.
2. **Lösung planen:** Welche Schritte und Daten werden benötigt? (Algorithmus)
   Programmieren findet zu 80 % im Kopf (oder auf Papier) und nur zu 20 % beim Tippen statt.
3. **Code schreiben:** so wenig wie möglich Code**!**
   KISS Keep It Simple, Stupid
   YAGNI „You Aren't Gonna Need It" = „Du wirst es nicht brauchen"
4. **Testen und Fehler suchen:** Debugging
   Nicht nur den Standardfall testen, sondern auch Extremfälle. Denke an die Benutzer, die Eingabefehler verursachen können.
5. **Bereitstellung/Rollout:** Benutzer schulen
6. **Verbessern und pflegen**

[4] vgl. IBM software-development

Woraus besteht ein Programm:

- **Deklaration:** Ankündigung der Existenz eines Elements (z. B. einer Variable, einer Konstante oder einer Funktion) Es wird ein Speicherplatz und ein Name vergeben sowie ein Datentyp festgelegt. Konstante verändern während des Programms ihren Wert nicht (z.B. mwst=0.19) Bei Variablen ändert sich der Wert während des Programms. Besondere Form sind Laufvariablen (z.B. i+=)
- **Wertzuweisung:** ein Variable bekommt einen bestimmten Wert (Inhalt)
  Eine Wertzuweisung ist keine Gleichung! **a=30 ✓ 30=a ✘**
- **Bedingungen:** Vergleiche (JA/NEIN) oder Fallauswahl; Entscheidungen,
  z. B. „wenn alter ≥ 18, dann …"
- **Schleifen:** Wiederholungen, abweisende Schleifen, abbrechende Schleifen Zählschleifen, z. B. „für jede Zeile in der Datei …"
- **Funktionen:** wiederverwendbare Bausteine, die eine Teilaufgabe lösen
- **Datenstrukturen:** Listen, Tabellen und Ähnliches, um Daten zu organisieren

Empfehlung für Bezeichnern (Namen):

- **Sprechende Namen wählen (Lesbarkeit vor Kürze):** z.B. summe_auftrag, kundenname,bestelldatum Für Laufvariable: x,i,n
- **Einschränkungen der Programmiersprache beachten:**
  keine Umlaute, ß, im Name kein / , % * usw.
- **Keine Datentypen im Namen verwenden:** z.B. alter_int, groeße_float.
- **Konstanten hervorheben:** Konstanten werden oft komplett in Großbuchstaben mit Unterstrichen geschrieben (z. B. MWST_SATZ).

Bug: (Käfer, Wanze, Fliegen): steht für Hard- oder Softwarefehler
Computer der 1. Generation (z.B. Z 3) arbeiteten mit Relais, in die sich Käfer, Fliegen etc. verirrten und die Kontakte blockierten, was den Programmablauf störte. Mit einem Debugger wird schrittweise der Code auf mögliche Fehler untersucht.[5] → Debugger

Compiler: Sind Programme, die eine Quell-Programmiersprache (z.B. PASCAL, C) in ein äquivalentes Programm einer Ziel-Programmiersprache (Maschinensprache) übersetzen. Compiler übersetzen das ganze Programm vor dem Start. Das übersetzte Programm kann

[5] vgl. Fischer und Hofer 2011

danach meist direkt gestartet werden. Viele Fehler werden beim Übersetzen erkannt, bevor das Programm läuft.

Debugger: Während einige Fehler als typografische oder syntaktische Fehler auftreten können, kann jeder Code, der die beabsichtigte Funktion der Software daran hindert, wie vorgesehen zu arbeiten, als Fehler betrachtet werden. Laufzeitfehler, Syntaxfehler und semantische Fehler können zu Abstürzen, falschen oder ungenauen Outputs, Sicherheitslücken und Datenverlust führen. Der Debugger soll dabei helfen, die Logikfehler aufzudecken.

Fehler-Arten:

- ***Compilerfehler***: man hat gegen die Anforderungen des Quelltextes verstoßen, der Compiler meldet den Fehler an einer Stelle im Skript, wo der Fehler sich auswirkt! → Syntaxfehler
- ***Laufzeitfehler***: man hat gegen die Anforderungen des Laufes von Programmen verstoßen. Auch der Computer kennt keine Division durch Null. Er bricht bei einer solchen einfach die Arbeit ab, um zu verhindern, dass das Programm falsch weiterarbeitet. → Semantikfehler
- ***Logische Fehler***: Im Programm steckt in diesem Falle ein Fehler, der nur sehr schwer bemerkt wird, denn das Programm arbeitet und bekommt bei Anwendung ein Ergebnis. Bei Nachkontrolle aber stellt sich heraus, dass diese Ergebnisse falsch sind. Ursache sind z.B. Fehler in der Anordnung der Formeln. Wenn, statt zu multiplizieren z.B. addiert wird. → Semantikfehler

Höhere Programmiersprache: für den Menschen verständlich, da meist englische Wörter verwendet werden, höherer Abstraktionsgrad (z.B. Python, PASCAL)

Integrierter Entwicklungsumgebungen (IDE): Eine IDE stellt auf ihrer Oberfläche alle Werkzeuge zur Verfügung, die zur Erstellung und Erprobung von Programmen benötigt werden.[6] Dazu zählen i.d.R. ein Quellcode-Editor, Interpreter, Compiler, Debugger in einer einzigen graphischen Benutzeroberfläche.[7] (z.B. Thonny, PyCharm, Code::Blocks, VS Code)

Interpreter: Liest und führt den Code beim Start schrittweise (nacheinander) sofort aus. Dabei findet ein ständiger Wechsel zwischen Programm und Interpreter statt. Für jede Programmausführung wird der Interpreter benötigt, was das Programm langsam macht. Fehler zeigen sich oft erst, wenn die betreffende Codezeile ausgeführt wird.

[6] Bär 2011, S. 71

[7] Vgl. Schneider und Smallay

Python verwendet intern eine Technologie, bei der eine Art kompilierter Code ausgegeben wird. Hierbei spricht man von Bytecode, der für den Interpreter aufbereitet ist, um schnellere Ausführungen zu erlauben. Dieser wird bei der ersten Ausführung eines Skripts erstellt und nur bei einer Veränderung des Skripts neu erstellt. So können Geschwindigkeitsvorteile erzielt werden, wenn unveränderte Skripte erneut ausgeführt werden.

Proprietäres Betriebssystem: wird vom Hersteller mitgeliefert und ist auf eine spezielle Hardware zugeschnitten, soll nicht auf andere Hardware portiert (übertragen) werden

Quellcode (Skript) ein vom Menschen geschriebene Text eines Computerprogramms. Er ist in einer bestimmten Programmiersprache verfasst (z.B. Python oder C) und besteht aus für Menschen lesbaren Wörtern, Zahlen, Symbolen und Logikbefehlen.

Semantikfehler: Logikfehler, der Ausdruck ist syntaktisch richtig, Programm bringt nicht das gewünscht Ergebnis. (z.B. geb=2000, aktu=2026, alter=geb-aktu oder x=summe/0) Logikfehler werden nur durch Testläufe erkannt. Hierbei hilft ein Debugger[8], der das Programm Zeile für Zeile ausführt und dabei den aktuellen Wert der Variablen anzeigt.

Syntaxfehler: Verstoß gegen die Schreibregeln, wird i.d.R. vom Interpreter bzw. Compiler erkannt z.B. print:“Ergebnis“ #richtig print(“Ergebnis“)

Systemnahe Programmiersprachen: Sprachen, mit denen man Programme schreiben kann, die eng mit der Hardware und dem Betriebssystem zusammenarbeiten (computernahe Programmiersprachen z.B. C). Eignen sich für Betriebssysteme, Gerätetreiber und Programme, bei denen Geschwindigkeit oder genauer Speicherzugriff wichtig sind. Verlangen umfassendes technisches Wissen.

Quellenverzeichnis

Bär, Bischofberger, Dehler et all. *Informatik und Informationstechnik.* Haan-Gruiten: Europa-Lehrmittel, 2011.

Fischer, Peter, und Peter Hofer. *Lexikon der Informatik.* Heidelber, London, New York: Springer, 2011.

Horn, Kerner, Forbig. *Lehr- und Übungsbuch Informatik, Band 1.* Wien: Carl Hanser, 2001.

Schneider, Josh, und Ian Smallay. *Was ist eine Integrierte Entwicklungsumgebung.* o.A.. o.A. o.A. https://www.ibm.com/de-de/think/topics/integrated-development-environment (Zugriff am 03. 11 2026).

[8] die Funktion wird bei manchen Programmiersprachen vom Interpreter übernommen