# Zweitveröffentlichung

**Sinz, Elmar J.**

## Programmiersprachen

Datum der Zweitveröffentlichung: 07.03.2024

Verlagsversion (Version of Record), Beitrag in Sammelwerk

Persistenter Identifikator: urn:nbn:de:bvb:473-irb-9393

**Erstveröffentlichung**

Sinz, Elmar J. (1998): „Programmiersprachen“. In: Wolfgang Lück (Hrsg.), Lexikon der Rechnungslegung und Abschlußprüfung, 4., völlig neu bearb. Aufl., München ; Wien: Oldenbourg, S. 603, doi: 10.1515/9783486795783.

**Programmiersprachen**

Künstliche, formale Sprachen zur Erstellung von Programmen (→ Software). Programme sind Handlungsanweisungen, die von Rechnersystemen ausgeführt werden können. Anwendungsprogramme stellen Lösungsverfahren für automatisierbare betriebliche Aufgaben dar. Systemprogramme realisieren anwendungsneutrale Funktionen eines Rechnersystems.

Der Prozessor eines Rechnersystems führt Programme aus, die in einer für den jeweiligen Prozessortyp gültigen, maschinencodierten Form vorliegen (Sprache der 1. Generation, 1st generation language, 1GL). Da die Erstellung maschinencodierter Programme durch den Menschen unhandlich, komplex und fehlerträchtig ist, werden symbolische Programmiersprachen verwendet. Ein in einer symbolischen Programmiersprache vorliegendes Programm wird entweder mit Hilfe eines Übersetzungsprogramms (Compiler) vollständig – ggf. über eine oder mehrere Zwischensprachen – in maschinencodierte Form überführt oder mit Hilfe eines Interpretationsprogramms (Interpreter) abschnittweise durch maschinencodierte Programmstücke ersetzt, die dann unmittelbar ausgeführt werden.

Unter dem Blickwinkel der Problemnähe lassen sich maschinenorientierte Assemblersprachen (2GL) und problemorientierte höhere Programmiersprachen unterschieden. Bei den Assemblersprachen korrespondiert ein symbolischer Befehl im allgemeinen mit einem maschinencodierten Befehl. Höhere Programmiersprachen abstrahieren vom Befehlsvorrat realer Prozessoren und bieten einen Sprachvorrat an, der näher an einer bestimmten Problemklasse orientiert ist. Beispiele für höhere Programmiersprachen der 3. Generation (3GL) sind Fortran, PL/I, Cobol, Pascal und C.

Hinsichtlich der Art der Programmbeschreibung lassen sich imperative und deklarative Programmiersprachen unterscheiden. Ein imperatives Programm spezifiziert eine Problemlösung in Form einer Befehlsfolge *(Wie)*. Beispiele für imperative Programmiersprachen sind die genannten höheren Programmiersprachen. Ein deklaratives Programm spezifiziert eine Problembeschreibung *(Was)*. Die zugehörige Befehlsfolge wird von einem weiteren Programm, welches als Inferenzmaschine bezeichnet wird, generiert. Beispiele für deklarative Programmiersprachen sind die Datenbanksprache SQL (4GL) und die Logikprogrammiersprache PROLOG (5GL).

Zunehmende Bedeutung erlangen das Paradigma der Objektorientierung und damit objektorientierte Programmiersprachen Ein Programm besteht dabei aus einer Menge von Objekten, die mit Hilfe von Nachrichten interagieren. Objekte sind zu Klassen zusammengefaßt. Das Konzept der Vererbung erlaubt es, Klassendefinitionen durch Erweiterung der Definitionen einfacherer Klassen zu erstellen.

Elmar J. Sinz