Programmiersprachen im Informatikunterricht

Werkzeuge zur Codierung von Algorithmen

Dr. Henry Herper

Otto-von-Guericke-Universität Magdeburg
Institut für Simulation und Graphik

Lisa-Weiterbildung - Aug. 2002
(angepasst Aug. 2004)

Fragen zum Informatikunterricht

- **Ist Informatikunterricht gleich Programmierunterricht?**
- **Ist das Programmieren allgemeinbildend oder nicht?**
- **Kann ein guter Informatikunterricht ohne Vermittlung von Programmiersprachen durchgeführt werden?**

ODER

Macht es Sinn, am Gymnasium wertvolle Unterrichtszeit für das Erlernen einer speziellen Programmiersprache mit allen ihren technischen Eigenheiten und Zufälligkeiten zu verbrauchen?

Allgemeinbildung

Kenntnisse und Fertigkeiten, die

- nicht auf einen bestimmten Beruf ausgerichtet sind,
- nicht automatisch, nebenher erworben werden,
- eine gewisse Universalität aufweisen,
- zum kritischen Vernunftgebrauch und zum
- verantwortlichen Umgang mit den erworbenen Kompetenzen anregen
- exemplarisch für eine allgemeinere Erkenntnis sind.

/Bussmann/Heymann 1987/

Programmieren? - NEIN

Die Konstruktion von Software – im Großen wie im Kleinen – gehört zum ingenieurwissenschaftlichen Teil der Informatik und ist nicht allgemeinbildend.

Programmierunterricht fördert, aber nur in sehr unbedeutendem Maße, das Verständnis für Probleme der Informationsverarbeitung.

/Burkert 1995/

Programmieren? - JA

- Ebenso, wie das elementare Rechnen die „Primärerfahrung“ der Mathematik ist, gilt dies entsprechend für das Programmieren als Primärerfahrung der Informatik.
- Die zentralen Begriffe der Informatik ... erwachsen aus den Erfordernissen des Programmierens. ...
- Programmieren ist Ausgangspunkt und Endpunkt gedanklicher Abstraktionsprozesse der Informatik. ...
- die Einführung des Programmierens spielt eine Schlüsselrolle für das Verständnis informatischer Grundbegriffe.

/Hoppe u. Luther 1996/

Programmieren? Weitere Informationen und Argumente

Peter Hubwieser
Programmieren im allgemeinbildenden Informatikunterricht am Gymnasium

http://ddi.in.tum.de/Hubwieser/Vortraege/2001.html

Historischer Abriss

1976 – Pascal/ELAN

1976 – „Empfehlungen, Ergebnisse, Abschlußbericht und Stellungnahme zu Basic“ – Prof. Klaus Haefner

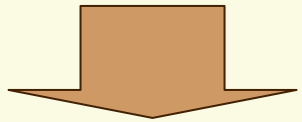
> **„BASIC ist für die geistige Entwicklung der Schülerinnen und Schüler nichts Vernünftiges und empfiehlt Pascal und ELAN (ausschließlich)“**

„Die bundesrepublikanische Welt sprach Pascal und BASIC wurde etwas für geistig Minderbemittelte“

/Baumann, Körber LOGIN 21 (2001) Heft 2/

Kriterien zur Auswahl von Programmiersprachen

Welches adäquate Werkzeug kann zur Lösung meines Problems verwendet werden?



Schaffung einer Familie von Programmiersprachen mit klaren semantischen Konzepten für unterschiedliche Anforderungen

Welches Werkzeug muss genommen werden, um alle Probleme zu lösen?

Einbeziehung aller bekannten Konzepte in eine Programmiersprache

Struktur der Rahmenrichtlinie

SJG 10	SJG 11	SJG 12
Grundlagen der Informationstechnik Projektarbeit unter Nutzung von Standardsoftware	Algorithmen-strukturen und ihre Implementierung	Wahlthema
Informatik und Gesellschaft Computer-Netzwerke	Datenstrukturen Informatisches Modellieren	Projektarbeit zur Software-entwicklung

Programmierfertigkeiten in der Rahmenrichtlinie (Entwurf 01/03)

SJG 10	SJG 11	SJG 12
Grundlagen der Informationstechnik Projektarbeit unter Nutzung von Standardsoftware	**Algorithmen-strukturen und ihre Implementierung**	**Wahlthema**
Informatik und Gesellschaft Computer-Netzwerke	**Datenstrukturen** **Informatisches Modellieren**	**Projektarbeit zur Software-entwicklung**

Programmierung im Bereich der Wahlthemen im SJG 12/1

(1) Modellbildung und Simulation
(2) Analyse und Design eines Informatiksystems
(3) Computergraphik
(4) Abstrakte Datentypen und ihre Implementierung
(5) Suchen und Sortieren von Daten
(6) Endliche Automaten und formale Sprachen
(7) Kryptologie
(8) Datenbankanwendungen zur dynamischen Webseitengenerierung

Ziele der RRL für die SJG 11/1

Algorithmen und ihre Implementierung

Die Schülerinnen und Schüler

- kennen einen Algorithmusbegriff und Eigenschaften von Algorithmen,
- können Algorithmen verbal formulieren und formal darstellen,
- kennen den Begriff der Variablen in der Informatik.

Ziele der RRL für die SJG 11/1

Die Schülerinnen und Schüler

• können Algorithmen analysieren, interpretieren und **in einer Programmiersprache codieren**,

• sind in der Lage, **Programmabschnitte zu strukturieren** und zu entsprechenden Einheiten zusammenzufassen,

• können **Programmoberflächen gestalten** und die **Eigenschaften der visuellen Komponenten bearbeiten** und ihre **Methoden (Ereignisse) mit entsprechendem Quelltext versehen**,

• kennen **grafische Basisroutinen** der verwendeten Programmiersprache und können diese anwenden.

Einzuführende Begriffe und Methoden (SJG 11/1)

- Variablen und ihre Verwaltung und interne Darstellung
- Datentypen (abstrakt und implementationsabhängig)
- Implementierung der algorithmischen Basisbausteine
Wertzuweisung, Folge, Verzweigung, Schleifen
- Ein- und Ausgabe von Daten
- Begriffe Klasse, Objekt, Eigenschaft, Methode, Ereignis
- Unterprogramme und Parameterübergaben
- Wert- und Referenzparameter

Ziele der RRL für die SJG 11/2

Datenstrukturen

Die Schülerinnen und Schüler

- kennen Strukturen zur **Verwaltung komplexer Daten** und zur rechnerinternen Repräsentation,
- können Daten in **Dateien** verschiedenen Typs verwalten.

Einzuführende Begriffe und Methoden (SJG 11/2)

- Reihung (Felder)
- Datenverbund (Record)
- Dateien
- Operationen auf Dateien

Wahlthema 3: Computergrafik

- Implementierung von Algorithmen zur 2D- Transformation (Translation, Skalierung, Rotation, Komposition) in der gewählten Programmiersprache
- Modellierung und Darstellung virtueller Welten mit einer geeigneten Sprache

Wahlthema 4: Abstrakte Datentypen und ihre Implementierung

- Implementierung von einfach verketteten Listen, doppelt verketteten Listen und Ringlisten durch dynamische Datenstrukturen oder Felder
- Implementierung der Grundfunktionen auf Listen (Initialisierung, Einketten, Ausketten, Suchen, Speichern in Dateien, Laden aus Dateien)
- Implementierung der Sonderformen Stack (Stapel) und Queue (Warteschlange)

Wahlthema 5: Suchen und Sortieren von Daten

- Implementierung von iterativen und rekursiven Schleifen
- Implementierung von elementaren Sortieralgorithmen (Selectionsort, Insertionsort, Bubblesort)
- Implementierung von einem Devide and Conquer Sortierverfahren (Quicksort)
- Implementierung von Suchverfahren
- Vergleich von iterativen und rekursiven Implementierungen am Beispiel des binären Suchens

Wahlthema 7: Kryptologie

- Implementierung verschiedener Verfahren der Kryptographie und der Kryptoanalyse,
- Chiffrieren von Klartexten nach verschiedenen Algorithmen in der gewählten Programmiersprache,
- Implementierung von Algorithmen zur Analyse und Dechiffrierung von fremden, verschlüsselten Texten.

Begriffe zu Sprachen

SPRACHE:
Eine Sprache über einem Alphabet ist eine beliebige Menge von Wörtern über diesem Alphabet. Die Wörter sind dabei Zeichenketten, die durch Aneinanderreihen von Zeichen des Alphabets entstehen.

SYNTAX:
Die Syntax einer Sprache ist die Menge aller Regeln, nach denen zuläs-sige Sätze in dieser Sprache gebildet werden können. Sie beschreibt die Struktur der Sprache und entspricht damit der Grammatik einer natürli-chen Sprache.

SEMANTIK:
Die Semantik ist die Lehre von der Bedeutung einer Sprache. Bei einer Programmiersprache beschreibt die Semantik, was während der Ausführung eines Programms oder eines Programmteils geschieht. Dabei werden auch die Wechselwirkungen berücksichtigt.

Programmiersprachen

Eine **Programmiersprache** ist ein notationelles System zur Beschreibung von Berechnungen in durch Maschinen und Menschen lesbarer Form.

- **Berechnung:** Nachweis der Berechenbarkeit z.B. durch die Turing-Maschine (Churchsche These)
- **Maschinenlesbarkeit:** Es muss eine eindeutige Übersetzung mit einer vertretbaren Komplexität möglich sein.
- **Lesbarkeit durch den Menschen**

Programmiersprachen

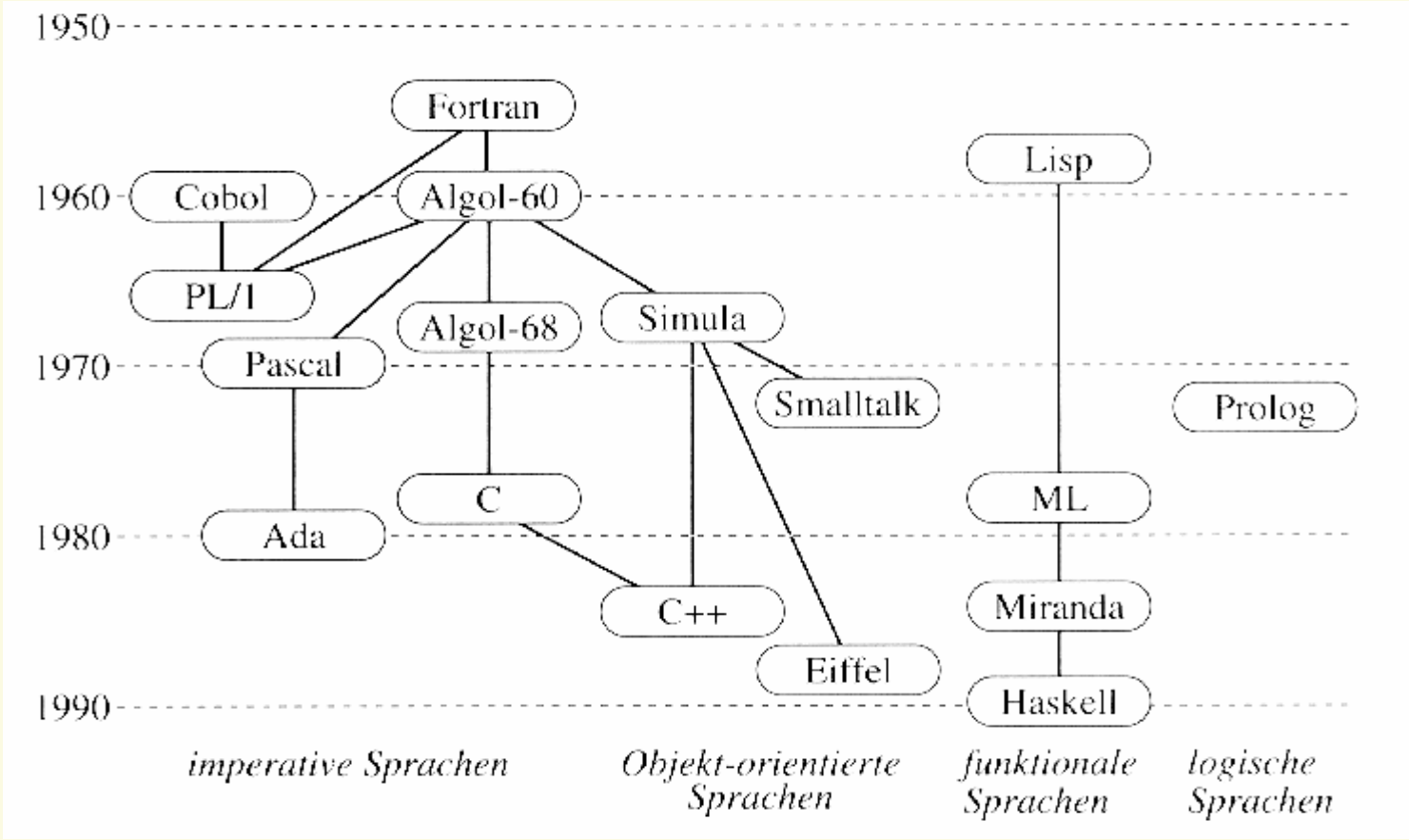
Eine Programmiersprache ist eine Sprache zur Formulierung von Algorithmen und Datenstrukturen für die Abarbeitung auf einem Computer.

Je nach dem Grad, mit der die Hardware bei der Programmierung beachtet werden muss, kann man Programmiersprachen in:

- Maschinensprachen,
- Assemblersprachen und
- höhere Programmiersprachen

untergliedern.

Entwicklungslinien von Programmiersprachen



Quelle: David A. Watt, Programmiersprachen, 1996

Didaktische Auswahl der Programmiersprachen

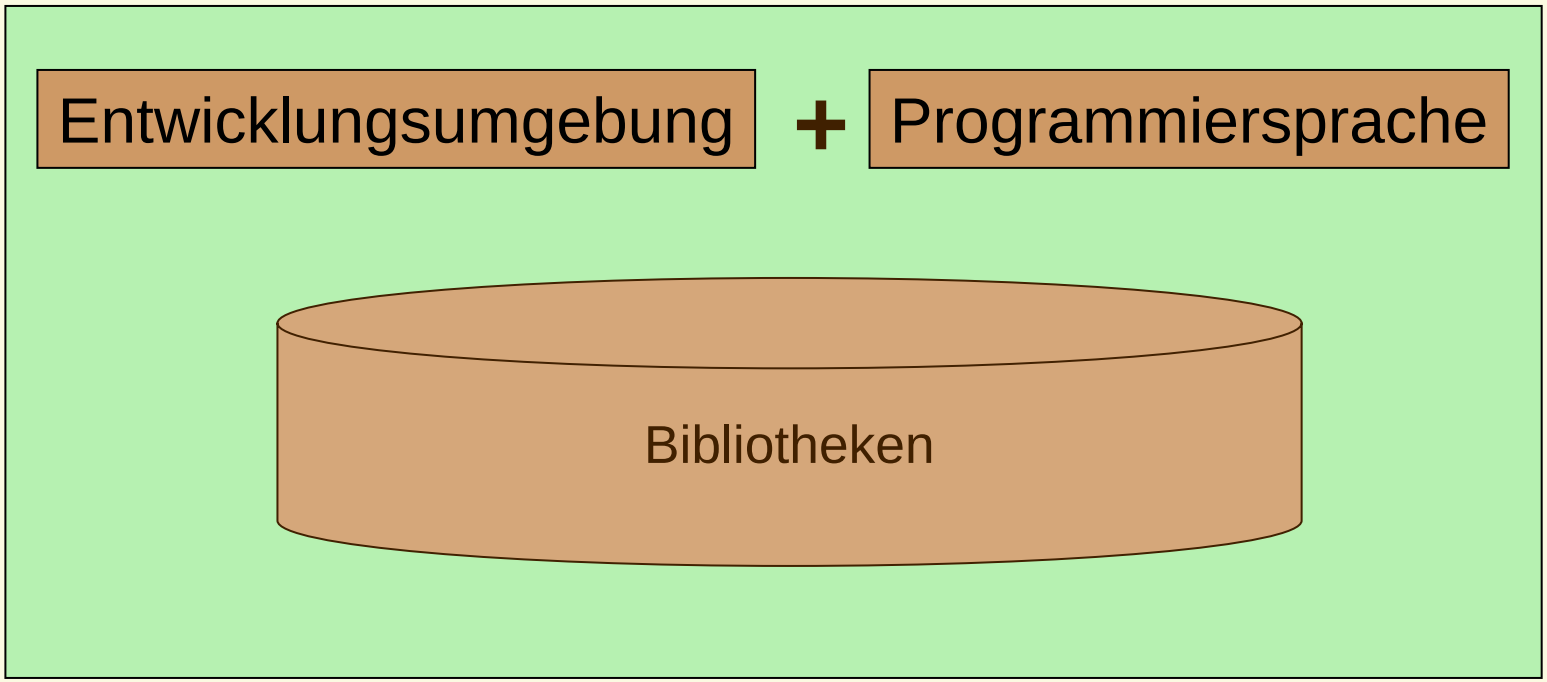
Professionelle Softwareentwicklungssysteme

- komplexe Systeme
- auf kommerzielle Softwareentwicklung ausgerichtet
- weit verbreitet
- Ausbildung an einem System, welches in der weiteren Ausbildung bzw. in der beruflichen Tätigkeit eingesetzt werden kann

Spezielle Ausbildungssprachen

- kleine Systeme
- didaktisch auf die Vermittlung der algorithmischen Grundstrukturen und informatischen Denkweisen optimiert
- einheitliches Ausgangsniveau
- für den praktischen Einsatz müssen die erlernten Techniken und Methoden auf ein anderes Entwicklungssystem übertragen werden

Programmierwerkzeuge



Anforderungen an ein Programmierwerkzeug

- **unterschiedliche Modellierungsarten** sollen mit einem Werkzeug realisierbar sein
- **klarer und intuitiver Sprachaufbau** (Syntaxelemente sollen die damit verbundene Semantik deutlich machen)
- **klares, ausgereiftes Typenkonzept**
- **präzise Fehlermeldungen** und geeignete Debugginghilfsmittel
- **geeignetes Hilfesystem**
- **unterstützendes Entwicklungssystem**

Imperative und objektorientierte Programmiersprachen

Viele imperative (prozedurale) Programmiersprachen sind in den letzten Jahren mit objektorientierten Erweiterungen versehen worden.

Imperative und objektorientierte Programmiersprachen – Modula-2

Ende der 70er Jahre: MODULA-2

Anwendungsgebiet: Ausbildungssprache, Systementwicklungssprache
Die Programmiersprache MODULA-2 ist eine Weiterentwicklung von Pascal und sollte die Nachteile von Pascal beseitigen. Das Konzept der modularen Programmierung wurde weiter verbessert. Objektorientierte Erweiterungen sind integriert.
Die Modellierung paralleler Prozesse wird unterstützt.

Verschiedene Entwicklungsumgebungen und Compiler werden angeboten.

Imperative und objektorientierte Programmiersprachen – Oberon

Von Nicolas Wirth wurde Oberon seit 1985 als Sprache mit Basiskonstrukten für das objektorientierte Programmieren entwickelt.

Ein Ziel war einen möglichst kleinen und schnellen Compiler zu erhalten. Oberon unterstützt strukturiertes, modulares und objektorientiertes Programmieren. Die implementierten Kontrollstrukturen entsprechen denen von Pascal.

Es sind verschiedene Entwicklungsumgebungen im nichtkommerziellen Bereich frei verfügbar.

Die Weiterentwicklung von Oberon-2 ist Component Pascal, das als kommerzielle Version angeboten wird.

Imperative und objektorientierte Programmiersprachen – Pascal

1971: PASCAL

Anwendungsgebiet: Ausbildungssprache

Drei Gruppen von Datentypen sind zugelassen. Dies sind einfache Typen, Zeiger (Pointer) und strukturierte Typen. Es sind einfache und strukturierte Anweisungen (bestehen wieder aus Anweisungen) möglich.

Die Möglichkeiten der strukturierten Programmierung werden durch eine leistungsfähige Unterprogrammtechnik und die Möglichkeit des rekursiven Funktionsaufrufes unterstützt. Turbo-Pascal enthält seit Version 5.5 objektorientierte Erweiterungen.

letzte kommerzielle Versionen: Borland-Pascal 7 (wird nicht mehr angeboten)

Imperative und objektorientierte Programmiersprachen – Delphi

Anwendungsgebiet: Erstellung von Windows-Applikationen

Kennzeichnend für diese Programmiersprachen sind die Eigenschaften

- objektorientiert,
- ereignisgesteuert und
- visuell.

Da Delphi für Windows-Anwendungen entwickelt wurde, unterstützt es besonders den Entwurf von Nutzeroberflächen. Der Zugriff zu Windows erfolgt über die Windows-API. In Delphi wurden Datenbankschnittstellen integriert.

Aktuelle Version: Delphi 7 (Delphi 6 ist in der Personal Edition für die private Anwendung kostenlos), KYLIX für UNIX-Anwendungen

Nachteil: Die aufwendige graphische Oberflächengestaltung lenkt von der Vermittlung informatischer Grundkenntnisse ab.

Imperative und objektorientierte Programmiersprachen – Java

Anwendungsgebiet: plattformunabhängige Sprache in Netzanwendungen

Java ist eine einfache, konsequent objektorientierte, multithreaded, robuste, architekturneutrale, übertragbare und dynamische Programmiersprache.

Sie basiert auf der Programmiersprache C++, wurde aber wesentlich vereinfacht. Java-Programme werden in Bytecode compiliert und können damit ohne weitere Compilierung auf verschiedenen Computern ausgeführt werden, sofern es eine Java Virtual Machine dafür gibt.

Professionelle Entwicklungsumgebungen stehen mit dem „Java Development Kit“ von Sun und Visual J++ von Microsoft zur Verfügung.

Imperative und objektorientierte Programmiersprachen – C/C++

Anwendungsgebiet: Systemprogrammiersprache, Echtzeitprogrammierung

Die Sprache ist blockorientiert und erlaubt eine sehr kompakte Schreibweise. Ein C-Programm ist eine Menge von Funktionen, die einander aufrufen können.

Es wurde zur Version C++ weiterentwickelt. Damit steht ein Werkzeug für die Entwicklung großer Softwaresysteme zur Verfügung. C++ besitzt leistungsfähige Komponenten zur objektorientierten Programmierung.

Fehlermeldungen sind nur sehr grob und unkonkret.

Diese Sprachfamilie ist für den Anfangsunterricht nur bedingt geeignet.

Imperative und objektorientierte Programmiersprachen – Visual Basic

Anwendungsgebiet: Ausbildungssprache, Makrosprache in Applikationen

Die Nutzung von BASIC verleitet zur unstrukturierten Programmierung und zur Schaffung von unübersichtlichen Programmen. Damit ist BASIC für die schulische Grundausbildung nicht geeignet.

Mit der Integration von **Visual-Basic** als Erweiterungssprache in die Microsoft-Office-Produkte nimmt die Bedeutung stark zu. Die verwendete Version ist **Visual Basic for Applications**.

Die Sprache besitzt keinerlei Möglichkeiten zur Vereinbarung von Klassen und damit zur Realisierung objektorientierter Programmierung.

Applikationssprache VBA

VBA – Visual Basic for Applications

- Makrosprache des Microsoft-Officepaketes (Word, Excel, PowerPoint, Access)
- Einsatzgebiet: Erweiterung der Funktionalität der Standardsoftware und Entwicklung von Oberflächen
- Vorteile:
  - Ergänzung von Office-Anwendungen durch nutzbringende Erweiterungen,
  - leichte Einarbeitung
  - schnelle Anfangserfolge,
  - modernes Erscheinungsbild,
  - Programmierumgebung mit gutem Debugging- und Hilfesystem.

Funktionale Programmiersprachen

Bei funktionalen Sprachen wird ein Programm als eindeutige Abbildung (Funktion) aus der Menge der Eingabedaten auf die Menge der Ausgabedaten betrachtet und stellt sich als Zusammensetzung aufeinander bezogener, einfacherer Funktionen dar. Es besteht die Möglichkeit, dass sich ein Programm selbst aufrufen kann. Vertreter sind

LISP und LOGO.

Funktionale Programmiersprachen – Logo

LOGO (vom griechischen logos - Vernunft)

Anwendungsgebiet: Kinder im Vorschulalter spielend an den Computer heranzuführen

LOGO ist funktionsorientiert. Das bedeutet, dass auch komplizierte zusammengesetzte Datenstrukturen als Funktionswerte von Funktionen dargestellt werden können. Grundlegende Unterschiede zu anderen Programmiersprachen ergeben sich aus dem Speicherkonzept. Es werden Behälter statt Variablen verwendet und es gibt keine Typen. Das LOGO-System nutzt neben Text immer eine hochauflösende Grafik. Logo wird interpretativ abgearbeitet.

Für den Einsatz in der schulischen Grundausbildung ist diese Sprache nur bedingt geeignet, da ein Typenkonzept fehlt.

Logische Programmiersprachen – PROLOG

Bei logischen Sprachen ist ein Programm die Niederschrift von Fakten ("Prädikaten", Definitionen) und Regeln, womit der Computer neue Fakten gewinnt und das Problem gelöst wird. Ein Vertreter ist beispielsweise PROLOG.

PROLOG ist eine Programmiersprache, die auf die Lösung von Problemen mit Hilfe des Prädikatenkalküls ausgerichtet ist.

Anwendungsbereiche: Datenbankabfragen, mathematische Beweise

Für Prolog existiert kein Standard, jedoch wird die an der Universität Edinburgh entwickelte Version heute am häufigsten verwendet.

Prolog wird in der Regel unter Kontrolle eines Interpreters ausgeführt.