2. Programmiersprachen

2.1 Grundlagen

Eine **Programmiersprache** ist eine Sprache zur Formulierung von Algorithmen und Datenstrukturen für die Abarbeitung auf einem Computer. Im Gegensatz zu natürlichen Sprachen, wo ein Wort mehrere Bedeutungen haben kann, ist in einer Programmiersprache eindeutig festgelegt, welche Zeichenfolgen als Programm zugelassen sind und was diese Zeichenfolgen bewirken.

Maschinensprache: Sprache, deren Alphabet nur aus zwei Zeichen (0, 1) besteht und vom Computer direkt verarbeitet werden kann.

Assemblersprache: Maschinenorientierte Programmiersprache, die anstelle des Binärcodes leichter verständliche Symbole, Mnemonics genannt, verwendet (z.B.: ADD, MOVE, JUMP). Ein Assemblerprogramm ist auf einem Computer nicht mehr direkt ablauffähig, sondern muss erst in ein entsprechendes Maschinenprogramm übersetzt werden. Bei größeren Problemlösungen werden die Programme sehr umfangreich und damit wartungsunfreundlich. Daher werden Assemblersprachen hauptsächlich nur noch da, wo Programme und Programmsysteme schnell reagieren müssen, und für Teile des Betriebssystems eingesetzt.

Höhere Programmiersprachen sind problemorientierte Sprachen, die weitgehend unabhängig von der Hardware und der natürlichen Sprache etwas näher sind.

2.2 Einteilung der höheren Programmiersprachen

<table>
<tr><th colspan="2">imperative</th><th colspan="2">deklarative</th></tr>
<tr><td colspan="2">Ein Programm besteht aus einer Folge von Befehlen (Anweisungen) an den Computer.

Das Programm beschreibt den Lösungsweg für ein Problem.</td><td colspan="2">Ein Programm beschreibt die allgemeinen Eigenschaften von Objekten und ihren Beziehungen untereinander.
Das Programm beschreibt zunächst nur das Wissen zur Lösung des Problems.</td></tr>
<tr><th>unstrukturierte</th><th>strukturierte</th><th>funktionale</th><th>logische</th></tr>
<tr><td>Ein Programm besteht aus Zuweisungen, Sprüngen und Schleifen. Ein solches Programm ist direkt an die Arbeitsweise eines Computers angelehnt.</td><td>Ein Programm ist vorwiegend eine Zusammensetzung aus einzelnen Algorithmenstrukturen. Die Datentypen werden zu Beginn eines jeden Programms vereinbart. Umfangreiche Programme können aus einzelnen Teilen zusammengesetzt werden.</td><td>Ein Programm wird als eindeutige Abbildung (Funktion) aus der Menge der Eingabedaten auf die Menge der Ausgabedaten betrachtet.</td><td>Ein Programm ist die Niederschrift von Fakten und Regeln, womit der Computer neue Fakten gewinnt und das Problem gelöst wird.</td></tr>
<tr><td><u>Vertreter:</u>
BASIC, FORTRAN</td><td><u>Vertreter:</u>
PASCAL, C,
Object Pascal</td><td><u>Vertreter:</u>
LISP, LOGO,
SCHEME</td><td><u>Vertreter:</u>
PROLOG</td></tr>
</table>

2.3 Etwas Sprachwissenschaft[1]

Die **Syntax** einer Sprache regelt, in welcher Beziehung die Zeichen einer Sprache zueinander stehen können. Syntax dürfen wir soweit mit Grammatik gleichsetzen, wie wir die Hinzunahme von interpretatorischen Inhalten verbieten. Sprechen wir über Inhalte von Zeichen, so machen wir keine Aussage mehr über die Syntax sondern über die **Semantik** einer Sprache.
Ein Beispiel soll das Gesagte verdeutlichen: "das Haus"
Syntaktisch gesehen ist "das Haus" eine Aneinanderreihung von 8 Zeichen (7 Buchstaben und ein Leerzeichen). Dass das Leerzeichen zwei Wörter trennt, ist Bestandteil der Semantik.
Der in einer Programmiersprache wie etwa Java, C++, Delphi oder Pascal verfasste Quelltext muss syntaktisch korrekt sein, dass er vom Compiler in eine ausführbare Maschinensprache übersetzt werden kann. Der Compiler muss also vor dem eigentlichen Übersetzen einen Quelltext auf syntaktische Korrektheit überprüfen (syntaktische Fehler führen zu einem Compiler error). Tut das Programm schließlich das, was es soll, ist es semantisch korrekt.



Wenn wir eine Programmiersprache erlernen, beschäftigen wir uns hauptsächlich mit der Syntax. Wenn wir später Programme schreiben wollen, die eine reale Situation modellieren, müssen wir uns zusätzlich mit der Semantik auseinandersetzen.
Eine sehr einfache Sprache ist die Sprache des Frohlockens. "Sie ist leicht erlernbar, denn sie ist eine Teilmenge der bayrischen Sprache und somit keine Fremdsprache (?)". Frohlocken stammt aus einer Aufgabe des 9. Bundeswettbewerbs Informatik 1990. Wie Sätze in dieser Sprache gebildet werden, beschreiben sogenannte Syntaxdiagramme. Die Sprache Frohlocken ist vollständig durch zwei Diagramme beschrieben. Das erste ist so zu lesen, dass ein Satz dieser Sprache aus ein- oder mehrmaligem **hosianna singen** besteht; mehrfaches **hosianna singen** wird durch Kommata getrennt und das letzte **hosianna singen** muss mit einem Ausrufezeichen (**!**) abgeschlossen werden. Kommas (**,**)und Ausrufezeichen werden nicht weiter erklärt, sie sind sog. Atome der Sprache. **hosianna singen** ist dagegen weiter auflösbar. Nach welchen Regeln dies geschieht, beschreibt das zweite Syntaxdiagramm. Unschwer erkennt man weitere Atome der Sprache: **ha**, **l**, **le**, **lu**, **u** und **ja.** In welcher Anordnung (allgemeiner in welcher Beziehung) sie auftreten dürfen ist leicht zu erkennen.
Übertragen wir das hier gelernte auf eine Computersprache, dann sind die Atome die reservierten Wörter einer Sprache wie wir sie in unseren Programmen kennen gelernt haben, ohne ihre Bedeutung (das wäre eine Frage der Semantik)) zu kennen. Ein ganzes, syntaktisch korrektes Programm entspricht dann einem Satz von Frohlocken. Ein Compiler, der bei einer Übersetzung prüft, ob das Programm syntaktisch korrekt ist, macht also zunächst nichts anders als wir, wenn wir etwa **halleluja,hahahalleluuja!** dahingehend untersuchen, ob es ein gültiger Satz der Sprache Frohlocken ist. Den Teil eines Compilers, der diesen Job erfüllt, nennt man einen Parser.

Aufgabe: Überprüfen Sie, ob folgende Sätze gültige Sätze der Sprache „Frohlocken“ sind!

luja,luuja!	hahalleluja luja,luja!	halleluja!
Luja	luja!luuja!	hallelu,hallelu,halleluja!

[1] Quelle: www.pohlig.de\Unterricht\Inf2003\Tag5\5.1_SyntaxDeFrohlockenden_Bayernsyntax.htm