GRUNDLAGEN

MARKUS SZYSKA

WAS IST EIN PROGRAMM?

- eine Beschreibung der Lösung einer vorgegebenen Aufgabe in einer spezifischen Programmiersprache.
- Lösung kann aus einzelnen Bearbeitungsvorschriften bestehen (Algorithmus)
- ein Algorithmus muss die Verarbeitung nach endlich vielen Schritten beenden und einen eindeutigen Ablauf besitzen
- Ein Programm besteht aus Algorithmen und verarbeitet Daten

EVA-PRINZIP

- Datenfluss durch ein Programm nennt man EVA-Prinzip
- **E**ingabe
- **V**erarbeitung
- **A**usgabe

PROGRAMMIERSPRACHEN

- gehören zu den formalen Sprachen
- Werden nach ihrer historischen Entwicklung in Generationen unterschieden
- Erste Generation: Maschinensprachen
- Zweite Generation: Assemblersprachen
- Dritte Generation: Prozedurale Sprachen
- Vierte Generation: 4GL (Generation Language)
- Fünfte Generation: Künstliche Intelligenz

MASCHINENSPRACHEN

- Damit ein Computer Probleme lösen kann, muss ihm der Lösungsweg in einer ihm verständlichen Art und Weise mitgeteilt werden. Eine Maschinensprache erfüllt genau diese Bedingung.

Beschreibung	Sowohl die Operationen als auch die Daten werden vom Programmierer ausschließlich als Bitfolge aus Nullen und Einsen (vgl. Abschnitt 6.2) eingegeben. Eine übliche Operation ist z. B. der Transport von Daten aus dem Speicher in ein Register.
Nachteile	Für jeden Computer müssen die Maschinenbefehle neu entwickelt werden, da diese Sprache von den Eigenschaften der Hardware (Prozessoren) abhängig ist. Programme in Maschinensprache sind schwer lesbar und mit einem hohen Programmieraufwand verbunden. Deshalb wird diese Sprache heute kaum mehr direkt eingegeben (allerdings werden auch heute noch alle Befehle für Computer letztendlich in Maschinenbefehle umgewandelt).
Beispiel	00011010 0011 0100

ASSEMBLERSPRACHEN

Beschreibung	Assembler-Sprachen sind wie Maschinensprachen an bestimmte Prozessoren gebunden. Übersetzungsprogramme, die Assembler-Programme in Maschinencode umwandeln, werden ebenfalls als **Assembler** bezeichnet.
Einsatz	Assembler-Sprachen werden überwiegend zur Programmierung der Hardware oder für schnelle, zeitkritische Programme eingesetzt.
Vorteile	Im Vergleich zur Maschinensprache bieten Assembler-Sprachen dem Programmierer durch Operationskürzel, z. B. ADD für addieren, wesentliche Erleichterungen. Da Assembler-Sprachen auf die maschinenspezifischen Besonderheiten des jeweiligen Computers abgestimmt sind, verbrauchen die Programme im Allgemeinen weniger Speicherplatz und sind meist auch schneller als ein entsprechendes Programm in einer anderen Programmiersprache.
Besonderheit	Die einzelnen Befehle der Assembler-Sprachen verwenden direkt die internen Befehle des Prozessors. Wer eine Assembler-Sprache erlernt, erfährt dabei viel über die Arbeitsweise des jeweiligen Prozessortyps.
Beispiel	ADD ax, 10

PROZEDURALE SPRACHEN

- Maschinensprachen waren ungeeignet zur Erstellung komplexer Anwendungsprogramme und für Menschen schlecht lesbar
- sind weitgehend unabhängig von einem Computersystem
- Werden als höhere Programmiersprachen bezeichnet
- der Weg zur Problemlösung ist eine Reihe von Anweisungen, die der Reihe nach abgearbeitet werden
- Programmiersprache muss von einem Übersetzer(Compiler oder Interpreter) in Maschinencode übersetzt werden
- Beispiele: Cobol, Fortran, Pascal, Basic, C/C++, Java

4GL (GENERATION LANGUAGE)

- der Programmierer beschreibt lediglich, was das Programm leisten soll, ohne den genauen algorithmischen Weg anzugeben.
- Eine einzelne Anweisung löst eine ganze Folge von internen Einzelschritten aus.
- Die Erstellung eines Programms wird dadurch wesentlich erleichtert.
- Da bestimmte Folgen von Arbeitsschritten innerhalb einer Anweisung automatisch ausgeführt werden, hat der Programmierer kaum einen Einfluss auf die internen Abläufe.
- Beispiele: SQL, Natural, Symphony, Open Access

KÜNSTLICHE INTELLIGENZ

- Der Grundgedanke des Forschungsgebietes der künstlichen Intelligenz ist es, zu untersuchen, unter welchen Bedingungen Computer menschliche Verhaltensweisen, die auf Intelligenz beruhen, nachvollziehen können.

- Für diese Forschungszwecke sind einige spezielle Programmiersprachen entwickelt worden. Diese Sprachen gehören meist zu den logischen oder funktionalen Programmiersprachen

- Inzwischen umfasst dieses Gebiet mehrere Fachbereiche, beispielsweise die Robotik, die zur Entwicklung von Robotern und deren komplizierten Bewegungsabläufen geführt hat, die Wissensverarbeitung und die Spracherkennung.

- Beispiel: Prolog, Lisp, Smalltalk

SKRIPTSPRACHEN

- sind in der Regel von einfacherer Natur
- werden meist über einen Interpreter ausgeführt
- Beispiele: PHP, JavaScript

INTERPRETER

- Übersetzung des Quelltextes während der Laufzeit eines Programms oder Skripts
- Interpreter übersetzen zeilenweise, erst dann wird jede Zeile einer Syntaxprüfung unterzogen
- Sind die Befehle in der Zeile korrekt, werden sie in Maschinencode übersetzt
- Tritt ein Fehler auf, wird der Übersetzungsvorgang abgebrochen und eine Fehlermeldung ausgegeben
- Interpretierte Programme werden durch die Prüfung zur Laufzeit langsamer ausgeführt als vor der Laufzeit übersetzte Programme

COMPILER

- Übersetzung des Quelltextes vor der Laufzeit eines Programms
- der Vorgang wird kompilieren genannt
- jede Anweisung wird einer Syntaxprüfung unterzogen
- Im Fehlerfall wird der gesamte Vorgang des Übersetzens abgebrochen und der Fehler muss beseitigt werden, bevor das Zielprogramm erstellt werden kann.

ZWEISTUFIGE ÜBERSETZUNG

- Nachteil: kompilierte Programme sind maschinenabhängig, somit plattformabhängig(z.B. Windows, Linux)
- Programmcode wird erst in einen plattformunabhängigen Zwischencode übersetzt, sogenannten Bytecode
- Dieser Code wird mit plattformspezifischem Interpreter ausgeführt
- Java-Interpreter werden virtuelle Maschine genannt(JVM)