

Konzepte der Programmiersprachen

Lehrstuhl Prof. Plödereder

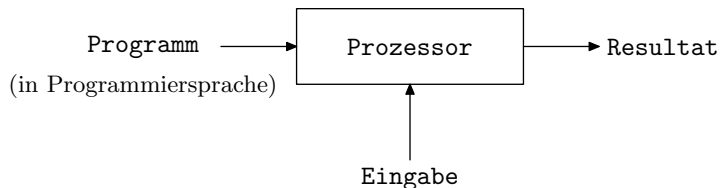
Eduard Wiebe

Institut für Softwaretechnologie
Abteilung Programmiersprachen und Übersetzerbau

Sommersemester 2007

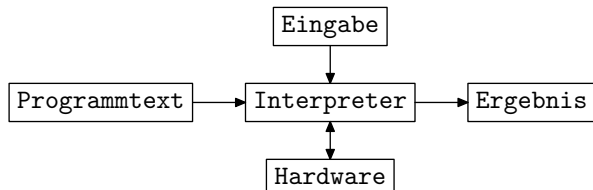
Programm-Ausführung

Programmiersprachen sind Hilfsmittel zur Beschreibung einer Vorschrift für einen Prozessor und bilden eine Schnittstelle zwischen Mensch und Rechner.



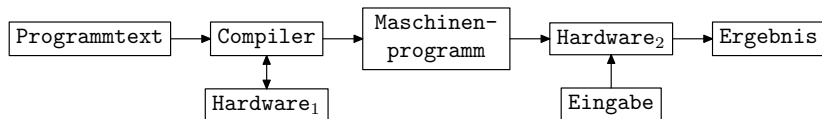
Interpreter

- ▶ Keine Erzeugung von Maschinencode
- ▶ Das Programm ist nicht eigenständig ausführbar
- ▶ Die Analyse des Programmtextes muss bei jeder Ausführung wiederholt werden. Keine Codeoptimierung möglich.
Folglich: zusätzlicher Bedarf an Rechenzeit.
- ▶ Dynamische Überprüfungen: zusätzliche Flexibilität in der Sprache, aber weniger Sicherheit
- ▶ Beispiele: Lisp, Prolog, Smalltalk, Perl, Python, ...



Compiler

- ▶ Analyse und Übersetzung des Programms findet nur einmal statt. Das generierte Maschinenprogramm kann anschließend beliebig oft ausgeführt werden.
- ▶ Mögliche Erhöhung der Sicherheit auf Kosten der Flexibilität: Durch statische Überprüfungen können zusätzliche Arten von Programmierfehlern gefunden werden.
- ▶ Typische Beispiele: Pascal, C, Fortran, Ada, ...



Das von Neuman-Modell

- ▶ Wesentliche Komponente: Speicher, Ablaufsteuerung, (arithmetische / logische) Verarbeitungswerke
- ▶ Daten und Befehle werden in einem gemeinsamen Speicher abgelegt.
- ▶ Ein Programm ist eine Folge von Befehlen, welche durch die Ablaufsteuerung sequenziell und schrittweise ausgeführt werden. Sprungbefehle dienen zur Umlenkung des Kontrollflusses. Ein Befehlszähler enthält die Adresse des jeweils auszuführenden Befehls.
- ▶ Die Inhalte der Speicherzellen können mit neuen Werten überschrieben werden.

Die von Neumann-*Sprachen* nehmen auf einer abstrakten Ebene auf dieses Modell Bezug:

- ▶ benannte Variablen sind symbolische Bezeichnungen für die Speicherzellen
- ▶ der Kontrollfluss (bzw. der Inhalt des Befehlszählers) kann durch Kontrollstrukturen (z. B. WHILE, IF) beeinflusst werden (Prinzip der Ablaufsteuerung durch Kontrollfluss)
- ▶ Beispiele: Fortran, C, Pascal, Ada, ...

Programmierung bedeutet in diesem Modell die Festlegung der

- ▶ Speicherbelegung (Entwicklung von Datenstrukturen, Variablendeklarationen)
- ▶ Speichertransformationen (Zuweisungen, Unterprogramme, Anweisungen)

Simulationsmodell

Idee: Programmierung = Simulation von realen Systemen (zum ersten Mal in Simula67 realisiert).

Aufgaben des Programmierers:

- ▶ Beschreibung der Bestandteile des realen Systems (Objekte)
- ▶ Beschreibung der Zusammenwirkens der Bestandteile (Nachrichten)

Die Ausführung besteht aus dem Austausch von Nachrichten zwischen Objekten und deren entsprechenden Reaktionen

Beispiele: Simula67, Smalltalk, Ruby

Funktionales Modell

- ▶ Ausführungsmodell: die Auswertung verschachtelter und im Allgemeinen rekursiver Funktionsaufrufe
- ▶ Alle programmiersprachlichen Konstrukte und alle vom Programmierer eingeführten Unterprogramme sind Funktionen, d. h. sie haben Parameter und liefern Werte zurück.
- ▶ Programmierung bedeutet die Entwicklung von Funktionen für die beabsichtigte Transformation von Eingabe- in Ausgabewerte
- ▶ Beispiele: Lisp, SML, Haskell, FP (**F**unctional **P**rogramming)

Datenfluss-Modell

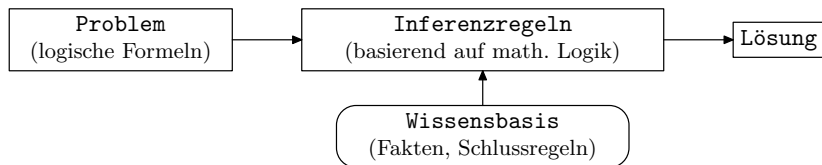
- ▶ Idee: jede Operation kann ausgeführt werden, sobald ihre Operanden ausgewertet sind (Prinzip der Ablaufsteuerung durch Datenfluss)
- ▶ Ausführungsmodell: Darstellung der Datenabhängigkeiten durch einen gerichteten Graphen (Abhängigkeitsgraph).
 - ▶ Knoten: die Operationen des Programms
 - ▶ Kanten: die Datenabhängigkeiten

Die Programmausführung besteht im Transport von Datenwerten entlang der Kanten. Sobald alle Operanden einer Operation vorliegen, wird diese ausgeführt und sendet ihr Ergebnis ab.

- ▶ Verzicht auf Kontrollfluss
- ▶ eignet sich gut zur Ausnutzung von Parallelität
- ▶ Beispiele: VAL, ID, SISAL

Logik-Modell

- ▶ Idee: Programmierung = Logische Herleitung einer Lösung aus einer Wissensbasis
- ▶ Aufgaben des Programmierers:
 - ▶ Erstellung der Wissensbasis
 - ▶ Formulierung des Problems
- ▶ Beispiele: Prolog, Concurrent Prolog



Gewünschte Eigenschaften

- ▶ Anpassung der Sprachen an die menschliche Denkweise und an die Struktur des zu lösenden Problems
- ▶ Unterstützung des Software Engineering
- ▶ effiziente Übersetzbarkeit in Maschinencode (Bezug zum Compilerbau)

Dies sind im Einzelnen:

1. problem-angepasste Ausdrucksform (writability)
2. gute Lesbarkeit (readability)
3. leichte Änderbarkeit (modifyability)
4. Verlässlichkeit (reliability)
5. Portabilität (portability)
6. Effizienz (efficiency)
7. Wartbarkeit (bedingt 2 und 3) (maintainability)

Pragmatische Forderungen:

- 8 : (leichte) Lernbarkeit der Programmiersprache
- 9 : kostengünstige Werkzeuge für PS

Eigenschaften von Programmiersprachen

1. Abstraktion
2. Einfachheit
3. Orthogonalität
4. Erweiterbarkeit, Flexibilität
5. Einfache Implementierung
6. Maschinen-Unabhängigkeit
7. Sicherheit (statische und dynamische Prüfungen)
8. Standard / Formale Definition
9. Prinzip der geringsten Verwunderung
10. ...

Vorsicht: Kriterien stehen z. T. im Konflikt zueinander!

Die Eigenschaften von Programmiersprachen haben sicherlich Einfluss auf das Erreichen der „...ilities“ und können die Entwickler implizit in diese Richtung lenken ...

...eine Garantie für die gewünschte Eigenschaft der produzierten Software können sie aber nicht geben.

Typische Software-Kostenverteilung

	Erstentwicklung	Gesamtkosten
Entwurf	40%	12%
Codierung	20%	6%
Test	40%	12%
Wartung	—	70%

⇒ Wartbarkeit hat höchste Priorität!

Folgerung

Programmiersprachen sind in erster Linie ein Hilfsmittel für die Kommunikation zwischen Projektmitgliedern und erst in zweiter Linie ein Kommunikationsmittel zwischen Mensch und Rechner.