Abstraktionsniveau von Programmiersprachen

1GL, 2GL, 3GL und 4GL im Überblick

Einführung

- Programmiersprachen werden nach ihrem Abstraktionsniveau klassifiziert
- Je höher die Generation, desto abstrakter und näher an der menschlichen Sprache
- Je niedriger die Generation, desto näher an der Maschinensprache
- Wir betrachten heute die vier Hauptgenerationen von Programmiersprachen

1GL - Maschinensprache

- Definition: Direkte Programmierung in Binärcode (0 und 1)
- Eigenschaften:
  - Niedrigstes Abstraktionsniveau
  - Direktes Ausführen durch den Prozessor
  - Keine Übersetzung nötig
- Beispiel:

  10110000 01100001
- Vorteile: Höchste Ausführungsgeschwindigkeit, direkter Hardwarezugriff
- Nachteile: Extrem komplex, fehleranfällig, hardwareabhängig

2GL - Assemblersprache

- Definition: Symbolische Darstellung von Maschinencode
- Eigenschaften:
  - Mnemonische(einprägsame) Befehle statt Binärcode
  - 1:1-Beziehung zu Maschinensprache
  - Benötigt Assembler zur Übersetzung
- Beispiel:

  MOV AL, 61h
  ADD AL, 1
- Vorteile: Bessere Lesbarkeit als 1GL, effiziente Ausführung
- Nachteile: Hardwareabhängig, komplex zu erlernen, zeitaufwändig

3GL - Höhere Programmiersprachen

- Definition: Problemorientierte Sprachen mit Abstraktion von Hardware
- Eigenschaften:
  - Näher an menschlicher Sprache
  - Hardwareunabhängig
  - Benötigt Compiler oder Interpreter
- Beispiele: C, C++, Java, Python
- Beispielcode (C):

```
cint x = 5;
int y = 10;
int sum = x + y;
```

- Vorteile: Portabilität, Lesbarkeit, produktivere Entwicklung
- Nachteile: Weniger Kontrolle über Hardware, potentiell langsamer als 2GL

4GL - Problemorientierte Sprachen

- Definition: Anwendungsorientierte Sprachen mit höchstem Abstraktionsniveau
- Eigenschaften:
  - Fokus auf das "Was" statt das "Wie„
  - Weniger Codezeilen für komplexe Aufgaben
  - Oft domänenspezifisch(Gebietsspezifisch)
- Beispiele: SQL, R, MATLAB, SAS
- Beispielcode (SQL):

```
SELECT name, age FROM users WHERE age > 18;
```

- Vorteile: Schnelle Entwicklung, Fokus auf Problemlösung
- Nachteile: Eingeschränkter Anwendungsbereich, weniger Flexibilität

Übersicht und Vergleich

Generation	Abstraktionsniveau	Beispielsprachen	Hauptanwendung
1GL	Sehr niedrig	Maschinencode	Direkte Hardwareprogrammierung
2GL	Niedrig	Assembly	Systemprogrammierung, Treiber
3GL	Mittel	C, Java, Python	Allgemeine Softwareentwicklung
4GL	Hoch	SQL, R, MATLAB	Datenverarbeitung, spezifische Domänen

Aktuelle Trends

- Entstehung von "5GL" - Programmiersprachen mit KI-Integration
- Low-Code und No-Code Plattformen als Evolution der 4

Fazit

- Jede Generation von Programmiersprachen hat ihre Vor- und Nachteile
- Höhere Generationen ermöglichen produktivere Entwicklung
- Niedrigere Generationen bieten mehr Kontrolle und Effizienz
- Die Wahl der richtigen Abstraktionsebene hängt von der Anwendung ab
- Moderne Entwicklung kombiniert oft verschiedene Abstraktionsebenen