

Hochschule Fulda

University of Applied Sciences



Programmiertechniken und Werkzeuge

Programmierparadigmen

Thomas Papendieck,
Senior-Consultant



- 1 Imperative Programmierung
- 2 Declarative Programmierung
- 3 Prozedurale Programmierung
- 4 Objektorientierte Programmierung
- 5 Funktionale Programmierung
- 6 typisierte Programmiersprachen
- 7 typenlose Programmiersprachen
- 8 Prinzipien der Programmierung

- 1.1 Definition
- 1.2 Beispiele
- 1.3 Ursprung
- 1.4 Vorteile



Imperative Programmierung (lateinisch imperare ‚anordnen‘, ‚befehlen‘) ist ein Programmierparadigma, nach dem „ein Programm aus einer Folge von Anweisungen besteht, die vorgeben, in welcher Reihenfolge was vom Computer getan werden soll“.

- Programmiersprachen
- Koch Rezepte
- Prozess-Checklisten

- Hardware benötigt "Schritt für Schritt"-Anweisungen
- erste Programmiersprachen waren sehr hardwarenahe

- Nähe zur Ausführungsebene
(Hardware)
- Programm ist aus sich selbst
vollständig nachvollziehbar

- 2.1 Definition
- 2.2 Beispiele
- 2.3 Vorteile



Die deklarative Programmierung ist ein Programmierparadigma, bei dem die Beschreibung des Problems im Vordergrund steht. Der Lösungsweg wird dann automatisch ermittelt.

Haskell, Lisp, Prolog, XAML
im weiteren Sinne auch SQL und XSLT.

- Die Programme sind oft kürzer als vergleichbare imperative Programme.
- Beweise (z. B. Korrektheitsbeweis, Beweise über Programmeigenschaften) sind dank einfacherer mathematischer Basis leichter durchführbar
- Es gibt keine Nebenwirkungen aufgrund der referentiellen Transparenz. Programme sind damit partiell auswertbar.

3.1 Definition

3.2 Beispiele



Die prozedurale Programmierung ergänzt das imperative Konzept aufeinander folgender Befehle um den Ansatz, einen Algorithmus in überschaubare Teile zu zerlegen. Je nach Programmiersprache werden diese Teile *Unterprogramm*, *Routine*, *Prozedur* oder *Funktion* genannt.

Die zweite Bedeutung ist als Sammelbegriff für die Programmierart, die bis in die 1990er Jahre lange vorherrschte, bis die *objektorientierte Programmierung* als Weiterentwicklung praxistauglich eingeführt wurde.

- Aufrufen von Unterprogrammen und das Durchleiten und die Rückgabe von Parameterwerten
- Hierarchie von Funktionen, sequentiell abgearbeitet
- Startpunkt des Programms in der Hauptprozedur
- Fehlerzustände über spezielle Rückgabewerte

Fortran, COBOL, ALGOL, C und Pascal

- 4.1 Definition
- 4.2 Beispiele
- 4.3 Vorteile



Bezieht sich auf eine Technik oder Programmiersprache, welche Objekte, Klassen und Vererbung unterstützt.

- Kapselung von (Zustands-) Daten und Methoden
- Polymorphie

- Simula-67
- Smalltalk
- C#, C++, Java, Kotlin, ...
-

- bessere Strukturierung
- Wiederverwendbarkeit von Programmteilen
- bessere Testbarkeit

- 5.1 Definition
- 5.2 Beispiele
- 5.3 Vorteile
- 5.4 Nachteile



Das Paradigma der funktionalen Programmierung ist eine Verfeinerung des deklarativen Paradigmas, bei dem die grundlegenden Ausdrücke die Erzeugung von Funktionen (die Abstraktion) und die Anwendung von Funktionen (die Applikation) sind.

- Funktionen \rightarrow eindeutige Abbildung einer Menge auf eine andere
- ineinander verschachtelte Funktionsaufrufe.
- Funktionen sind gegenüber allen anderen Datenobjekten gleichberechtigt \rightarrow Parameter von Funktionen und deren Rückgabewert
- Daten "fließen" durch das Programm

- Lisp, Haskell, OCaml, F#, Erlang, Clojure, Scala
- C++ (ab Version 11), Java (ab Version 8), Kotlin

- Berechnungen effizient und übersichtlich darstellbar
- Generierung neuer Funktionen zur Laufzeit
- prinzipbedingt für nebenläufige Prozesse geeignet

■ nur für Berechnungen

- 6.1 Definition
- 6.2 Beispiele
- 6.3 Vorteile
- 6.4 Nachteile



Für Variablen sowie Parameter und Rückgabewerte von Prozeduren wird festgelegt, von welchem Datentyp sie sind. Die Festlegung des Datentyps kann implizit oder explizit erfolgen.

■ C

■ C++

■ Java

■ go

■ C#

■ Python

■ Haskell

■ TypeScript

- fail fast
- Vervollständigungsvorschläge durch die IDE

- verwendete Typen müssen zum Programmierzeitpunkt bekannt sein
- nachträgliche Änderungen an Typen können bestehenden Code brechen

- 7.1 Definition
- 7.2 Beispiele
- 7.3 Vorteile
- 7.4 Nachteile



Der Typ von Variablen, Parametern
und Rückgabewerten wird nicht
festgelegt.

- sh
- cmd
- JavaScript
- x86 Assembler

- nur faktische Kompatibilität
- implizite Konvertierung

- Typenfehler treten erst zur Laufzeit des Programms auf

8.1 SOLID vs. STUPID

8.2 weitere Prinzipien





do's & don'ts

- Separations of Concern
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle
- Singleton
- Tight Coupling
- Untestability
- Premature Optimization
- Indescriptive Naming
- Duplication

KISS Keep It Simple (and) Stupid

FCoH Favor Composition over Inheritance

SLA Single Layer of Abstraction

YAGNI You Ain't Gonna Need It

IOC Inversion of Control

DI Dependency Injection