Hochschule Fulda

University of Applied Sciences

Programmiertechniken und Werkzeuge

Programmierparadigmen

Thomas Papendieck,
Lead–Consultant

Inhalte

Hochschule Fulda
University of Applied Sciences

Definition Programmierparadigma[1]

Ein Programmierparadigma ist ein fundamentaler Programmierstil.
Der Programmierung liegen je nach Design der einzelnen Programmiersprache verschiedene Prinzipien zugrunde.
Diese sollen den Entwickler bei der Erstellung von ‚gutem Code' unterstützen, in manchen Fällen sogar zu einer bestimmten Herangehensweise bei der Lösung von Problemen zwingen.

[1] `https://de.wikipedia.org/wiki/Programmierparadigma`

Hochschule Fulda
University of Applied Sciences

1

Imperative Programmierung - Definition

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".

Imperative Programmierung - Beispiele

- 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

Deklarative Programmierung - Definition

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

Deklarative Programmierung - Beispiele

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

Deklarative Programmierung - Vorteile

- 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

Prozedurale Programmierung - Definition

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.

Prozedurale Programmierung - Definition

- 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

Prozedurale Programmierung - Beispiele

`Fortran`, `COBOL`, `ALGOL`, `C`, `Pascal` …

4

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

Funktionale Programmierung - Definition

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.

Funktionale Programmierung - Definition

- Funktionen → eindeutige Abbildung einer Menge auf eine andere
- ineinander verschachtelte Funktionsaufrufe.
- Funktionen sind gegenüber allen anderen Datenobjekten gleichberechtigt → 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

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

typisierte Programmiersprachen - Beispiele

- C / C++
- Java
- go
- C#
- Haskell
- TypeScript

- fail fast
- Vervollständigungsvorschläge durch die IDE

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

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

typenlose Programmiersprachen - Beispiele

- `sh`
- `cmd`
- `JavaScript`
- `x86 Assembler`

- nur faktische Kompatibilität
- implizite Konvertierung

- Typenfehler treten erst zur Laufzeit des Programms auf

do's & don'ts

- **S**eparations of Concern
- **O**pen/Closed Principle
- **L**iskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion Principle

- **S**ingelton
- **T**ight Coupling
- **U**ntestability
- **P**remature Optimization
- **I**ndescriptive Naming
- **D**uplication

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