Programmierparadigmen

Imperative Programmierung

> Die imperative Programmierung (von lateinisch imperare = befehlen) ist das älteste Programmierparadigma. Gemäß diesem Paradigma besteht ein Programm aus einer klar definierten Abfolge von Handlungsanweisungen an einen Computer.

Der Quellcode imperativer Sprachen reiht also Befehle aneinander, die festlegen, was wann vom Computer zu tun ist, um ein gewünschtes Ergebnis zu erzielen. In Variablen eingesetzte Werte werden dabei zur Laufzeit des Programms verändert. Um die Befehle zu steuern, werden Kontrollstrukturen wie Schleifen oder Verzweigungen in den Code integriert.

Strukturierte Programmierung

Beim strukturierten Programmierungsansatz handelt es sich um eine **vereinfachte Form** der imperativen Programmierung. Die entscheidende Änderung zum Grundprinzip: Anstelle der absoluten Sprungbefehle (Anweisungen, die dazu führen, dass die Verarbeitung nicht mit dem nachfolgenden Befehl, sondern an anderer Stelle weitergeführt wird) sieht dieses Paradigma für die Software-Programmierung den **Einsatz von Kontrollschleifen bzw. -strukturen vor.**

Ein Beispiel hierfür ist die Nutzung von „**do…while**“, um eine Anweisung automatisch so lange auszuführen, wie eine bestimmte Bedingung wahr ist (mindestens ein Mal).

> Populäre Vertreter sind **C** und **C++, C#, COBOL, Fortran, Java, Pascal, Python** oder auch **Visual Basic**.

Prozedurale Programmierung

Das prozedurale Programmierparadigma erweitert den imperativen Ansatz um die Möglichkeit, **Algorithmen in überschaubare Teile aufzugliedern**. Diese werden als Prozeduren oder – je nach Programmiersprache – auch als Unterprogramme, Routinen oder Funktionen bezeichnet.

Sinn und Zweck dieser Aufteilung ist es, den **Programmcode übersichtlicher** zu machen und unnötige **Code-Wiederholungen zu vermeiden.** Durch die Abstraktion der Algorithmen stellt das prozedurale Software-Paradigma einen entscheidenden Schritt von einfachen Assemblersprachen hin zu komplexeren Hochsprachen dar.

>

> Beispiele für typische prozedurale Programmiersprachen sind **C und Pascal**.

Objektorientierte Programmierung

Objektorientierte Programmierung (OOP) ist ein Modell der Computerprogrammierung, bei dem das Softwaredesign auf **Daten oder Objekten basiert** und nicht auf Funktionen und Logik. Ein Objekt kann als ein **Datenfeld definiert** werden, das eindeutige Attribute und Verhaltensweisen aufweist.

Die Struktur beziehungsweise die Bausteine der objektorientierten Programmierung umfassen:

- **Klassen** sind benutzerdefinierte Datentypen, die als Blaupause für individuelle Objekte, Attribute und Methoden dienen.
- **Objekte** sind Instanzen einer Klasse, die mit speziell definierten Daten erstellt werden. Objekte können realen Objekten oder einem abstrakten Gebilde entsprechen. Bei der anfänglichen Definition einer Klasse ist die Beschreibung das einzige Objekt, das definiert wird.
- **Methoden** sind Funktionen, die innerhalb einer Klasse definiert sind und das Verhalten eines Objekts beschreiben.
- **Attribute** werden in der Klassenvorlage definiert und stellen den Zustand eines Objekts dar. Objekte haben Daten, die im Attributfeld gespeichert werden. Klassenattribute gehören zur Klasse selbst.

> „ Zu den Programmiersprachen, die hauptsächlich für die objektorientierter Programmierung entwickelt wurden, gehören **Java, Python und C++**

Deklarative Programmierung

> Kennzeichnend für die deklarativen Programmiersprachen ist, dass sie immer ein gewünschtes Endergebnis beschreiben, statt alle Arbeitsschritte aufzuzeigen. Um das Ziel zu erreichen, wird bei der deklarativen Programmierung der Lösungsweg automatisch ermittelt. Dies funktioniert so lange gut, wie die Spezifikationen des Endzustands klar definiert sind und ein passendes Ausführungsverfahren existiert. Trifft beides zu, ist die deklarative Programmierung sehr effizient.

Da die deklarative Programmierung das „Wie“ nicht festschreibt, sondern auf einem sehr hohen Abstraktionsniveau arbeitet, lässt das Programmierparadigma zudem Raum für Optimierung. Wird ein besseres Ausführungsverfahren entwickelt, lässt sich dies über den integrierten Algorithmus auffinden und verwenden. Auf diese Weise ist das Paradigma sehr zukunftssicher: Beim Schreiben des Codes muss das Verfahren, wie das Ergebnis zu erreichen ist, nicht feststehen.

Logische Programmierung

Das logische Software-Paradigma, das auch als prädikative Programmierung bezeichnet wird, **beruht auf der mathematischen Logik**. Anstelle einer Folge von Anweisungen enthält eine Software, die nach diesem Prinzip programmiert wird, eine Menge von Grundsätzen, die sich als Sammlung von Fakten und Annahmen verstehen lässt. Jegliche Anfragen an das Programm werden verarbeitet, indem der Interpreter auf diese Grundsätze zurückgreift und zuvor definierte Regeln auf diese anwendet, um zum gewünschten Ergebnis zu kommen.

> Einer der wichtigsten Vertreter der logischen Programmierung ist die Programmiersprache **Prolog**.

Funktionale Programmierung

Im Mittelpunkt der funktionalen Herangehensweise beim Programmieren stehen **Funktionen**. Bei einem funktionalen Programm können alle Elemente als Funktion aufgefasst und der Code kann durch **aneinandergereihte Funktionsaufrufe** ausgeführt werden. Im Umkehrschluss gibt es keine eigenständigen Zuweisungen von Werten.

> Zu den wichtigsten Programmiersprachen, die auf dem funktionalen Ansatz basieren, zählen **LISP, Haskell, F#** oder **Erlang**

Created 2022-10-27 07:01:51 UTC by Joshua Lieder
Updated 2022-10-28 16:45:41 UTC by Joshua Lieder