Lisp

Lisp (*list processor*) wurde auch in dieser Zeit (1958) entwickelt:
von John McCarthy, der FORTRAN um Listen erweitern wollte.

Lisp verbreitete sich vor allem im Bereich der künstlichen Intelligenz.

Lisp basiert auf dem mathematischen λ-**Kalkül** von Church und Kleene.

Daten und Programme werden einheitlich als Listen dargestellt
(S-Ausdrücke);
der Programmablauf wesentlich über Rekursion gesteuert.

Die vielen Klammern eines Lisp-Programms verursachten die inoffizielle Langform des Namens: *lots of irritating superflous parentheses*

Speicher für Listenelemente wird dynamisch zur Laufzeit bereitgestellt, nicht mehr benötigter Speicher durch *garbage collection* eingezogen.

Funktionen sind in Lisp „normale“ Daten.

Lisp

Berechnungen werden in Lisp nur durch Anwenden von Funktionen auf Daten ausgeführt; es gibt keine Variablen als Behälter von Werten.

Man spricht daher von funktionaler (applikativer) Programmierung.

Moderne Varianten einer funktionalen Sprache sind:
Scheme, Common Lisp, CLOS, ML, OCaml oder Haskell.

Die 1960er

BASIC (*beginners all-purpose symbolic instruction code*) wurde 1963 von Thomas E. Kurtz und John G. Kemeny vorgestellt als einfache Sprache für Studenten außerhalb des technisch-naturwissenschaftlichen Bereichs.

Weil BASIC zu Anfang nur interpretativ abgearbeitet wurde und relativ klein und einfach war, eignete sie sich gut für die ersten „Home Computer“.

SNOBOL (*string oriented symbolic language*) von David J. Farber, Ralph E. Griswold und Ivan P. Polonsky (AT&T Bell Labs) 1964 vorgestellt, war eine Sprache zur Verarbeitung von Zeichenketten.
Moderne Nachfolger sind SNOBOL 4 und ICON, aber auch Skriptsprachen wie Perl oder Tcl.

Die 1960er

PL/I (*programming language one*) wurde 1965 in der ersten Fassung als NPL (*new programming language*) bezeichnet.

Ziel war die Vereinigung der wichtigsten Konzepte von FORTRAN, COBOL und Algol 60.

Herausgekommen ist eine Anhäufung von sich z.T. widersprechenden „Features“, z.B. gibt es für Variablen wahlweise eine statische Speicherzuordnung (wie in FORTRAN) oder eine Blockstruktur (wie in Algol) oder aber eine dynamische Speicherzuordnung über explizite Aufrufe (wie auch in Pascal).

Ohne die Marktmacht von IBM wäre diese Sprache wohl kaum erfolgreich geworden.

Die 1960er

Simula 67 von Ole-Johan Dahl und Kristen Nygaard 1967 vorgestellt: Algol-ähnlich mit Erweiterungen für die diskrete Simulation.

Konzept der **Klasse** als Erweiterung des Blockkonzeptes:

- Eine Klasse besteht aus einer Menge von Datenvereinbarungen und Prozeduren.
- Ein **Objekt** einer Klasse wird zur Laufzeit erzeugt; Lebensdauer ist nicht an die Blockstruktur gebunden.

Simula legte damit die Grundlage zur objekt-orientierten Programmierung.

Diese Konzepte findet man in recht reiner Form in Smalltalk, oder –mit anderen Konzepten– in C++, Delphi, Eiffel, Java und Python.

Die 1960er

Algol 68 entstand durch systematische Verallgemeinerung von Algol 60.
Es sollten wenige, aber beliebig kombinierbare (orthogonale) Sprachkonzepte Verwendung finden.

Algol 68 spezifizierte die Typen der Parameter bereits im Funktionskopf (und nicht getrennt wie in Algol 60) und ermöglichte die Initialisierung von Variablen direkt bei der Deklaration (sogar für Felder).
Die Blockklammerung durch Schlüsselwörter
(z.B. **if. . . then. . . else. . . fi**, **for. . . do. . . od**)
ermöglicht auch eine eindeutige Zuordnung des else-Teils.

Diese Sprache hatte außerhalb der Universitäten nur geringe Akzeptanz, da die formale Definition über zweistufige Grammatiken festgelegt wurde und die Handbücher zur Sprache dadurch schwer lesbar waren.

TPK-Algorithmus in Algol 68

```
begin
    [0:10] real a;

    proc f = (real t) real :
        sqrt(abs(t))+5*t↑3;

    for i from 0 to 10 by 1 do read(a[i]) od;
    comment Laufvariable i ist implizit deklariert comment
    for i from 10 to 0 by -1 do
        real y := f(a[i]);
        if y > 400 then print((i,"too large")) else print((i,y))
        fi
    od
end
```

Die 1970er

Pascal war 1971 die „Antwort“ von Niklaus Wirth auf das seiner Meinung nach zu mächtige Algol 68.

Trotz vieler Einschränkungen in der Sprache wurde Pascal, speziell als Ausbildungssprache, sehr beliebt.

Abgelöst wurde Pascal Ende der 1980er Jahre durch Modula-2 und Oberon.

Für praktische Zwecke wurde die Sprache erst durch Erweiterungen wie in Borlands Turbo-Pascal brauchbar.

Eine objektorientierte Erweiterung von Pascal ist Delphi.

Die 1970er

Prolog (*programming in logic*) wurde 1972 von Alain Colmerauer entwickelt.

Prolog nutzt als nicht-prozedurale Sprache einen Teil der mathematischen Logik direkt als Programmiersprache.

Fakten und Regeln bilden eine Datenbasis, an die man Anfragen stellen kann.
Das System versucht dann, durch logisches Schließen die Anfrage aus der Datenbasis zu beantworten.

Prolog definiert einen speziellen Typ von Programmierung, die logische (deklarative) Programmierung, und ist die wichtigste Programmiersprache für „Expertensysteme“.

Die 1970er

C wurde 1972 von Dennis Ritchie geschaffen, um große Teile des UNIX-Betriebssystems in einer höheren Programmiersprache schreiben zu können.
Der Erfolg von C beruht in erster Linie auf dem Erfolg von UNIX.

C erlaubt eine relativ maschinennahe Programmierung, verleitet jedoch manchmal zur „Trickprogrammierung“.

Ende 1989 wurde C vom ANSI-Komitee standardisiert.
Das ANSI-C bietet dem Programmierer die gleiche „Sicherheit“ wie Pascal (z.B. durch Typprüfung), ohne ihm die Möglichkeiten der maschinennahen Programmierung zu nehmen.

Die 1970er

C++ entstand Anfang der 1980er Jahre als objekt-orientierte Weiterentwicklung von C (Bjarne Stroustrup).

In C bzw. C++ wird heute ein Großteil der Betriebssystem- und Anwendungssoftware im Personal-Computer und Workstation-Bereich geschrieben.

TPK-Algorithmus in C++

```
#include <iostream.h>
#include <math.h>

double f (double x) {
    return (sqrt(fabs(x)) + 5.0*pow(x,3.0));
}

int main (int argc, char** argv) {
    double A[11];
    for (int i=0; i<11; i++) { cin >> A[i]; }
    for (i=10; i>=0; i--) {
        double y = f(A[i]);
        if (y > 400.0) { cout << i << "TOO LARGE\n"; }
        else { cout << i << ' ' << y << '\n'; }
    }
    return 0;
}
```

Die 1970er

Smalltalk entstand Anfang der 1970er Jahre im Xerox Forschungslabor in Palo Alto im Rahmen des Dynabook Projekts.

Das Dynabook basierte auf einer Idee von Alan Kay für einen persönlichen Rechner, der eine –für die damalige Zeit– revolutionäre graphische Benutzeroberfläche anbot.

Informationen wurden in überlappenden Fenstern angezeigt und die Steuerung des Rechners geschah mit Hilfe einer Maus über Befehle, die man sich nicht mehr merken musste, da sie in Menüs angezeigt wurden.

Außerdem sollte das Dynabook Musik verarbeiten und Verbindungen zu anderen Geräten oder Datenbanken herstellen können.

Die 1970er

Das System wurde in einer neuen, rein objektorientierten Programmiersprache Smalltalk implementiert.

Die erste Version war Smalltalk 72 (1972),
die letzte, auch heute noch aktuelle Version ist Smalltalk 80 (Adele Goldberg, 1980).

Smalltalk-Programme werden von einem Compiler in einen Bytecode übersetzt und dieser wird dann von einem Interpretierer, der „Smalltalk Virtual Machine“, abgearbeitet.

Die durch dieses Projekt gewonnenen Erkenntnisse sind sehr(!) langsam in den „normalen“ Computermarkt geflossen. Viele haben von diesem Projekt „gelernt“, so `Apple` mit dem `Macintosh` (1983), `Microsoft` mit `Windows`, `Unix` mit `X-Windows`.

Die 1980er

Ada (nach: Ada King, Countess of Lovelace) erschien 1980 nach langer Konzeptionszeit als eine vom amerikanischen Verteidigungsministerium geforderte und initiierte Programmiersprache.

Wie bei PL/I wollte man eine Sprache für ein weites Anwendungsspektrum entwickeln.

Ada enthält neue Sprachkonzepte

- zur Kapselung von Daten (*package*),
- zur Parallelverarbeitung (*task*) und
- zur Ausnahmefall-Bearbeitung (*exception*)

und eignet sich damit gut zur Steuerung komplexer technischer Systeme.

Ada wird viel im militärisch-technischen Bereich benutzt.

Die 1980er

ML (*meta language*) entstand ab 1973 in Edinburgh als Zwischensprache zu Robin Milners ambitioniertem Projekt LCF (*logic for computable functions*) eines Theorem-Beweisers und vereint die Flexibilität einer funktionalen Sprache mit einem sehr ausgereiften Typ-System, das eine starke Typ-Prüfung erlaubt.

In ML müssen Typen nicht mehr deklariert werden. Stattdessen schließt dieses **Hindley-Milner-Typsystem** aus dem Gebrauch der Namen auf mögliche Typen und prüft, ob alle Vorkommen dieses Namens konsistent sind.

ML wurde 1986 standardisiert (Standard ML, SML).

TPK-Algorithmus in OCaml

Eine Variante von ML ist Caml (*categorical abstract machine* + ML), die um 1985 von Gérard Huet entwickelt wurde und
1990 von Xavier Leroy um objekt-orientierte Konzepte zu OCaml (*objective Caml*) erweitert wurde.

```
let tpk l =
    let f x = sqrt x +. 5.0 *. (x ** 3.0) in
    let p x = x < 400.0 in
        List.filter p (List.map f (List.rev l))
```

Die 1980er

Miranda wurde 1985 von David Turner als erste funktionale Sprache für den kommerziellen Gebrauch veröffentlicht.

Ein Miranda-Programm ist eine Menge von Gleichungen, die mathematische Funktionen und abstrakte Datentypen definieren. Die Reihenfolge der Gleichungen ist irrelevant.

Miranda nutzt den Call-By-Need.

Blöcke werden nicht geklammert, sondern durch Einrückung gekennzeichnet.

Turners Vorläufer von Miranda waren SASL (*St. Andrews Static Language*) und KRC (*Kent Recursive Calculator*).

Die 1990er

Haskell (nach Haskell B. Curry) entstand ab 1987 aus einer breiten Initiative für eine rein funktionale Programmiersprache mit verzögerter Auswertung.

Entwickler waren u.a. Paul R. Hudak, Philip Wadler und Simon Peyton Jones.

Die Sprache ist streng typgebunden, ohne dass die Typen deklariert werden müssen. Neu sind Typklassen.

Fallunterscheidungen in Funktionsdefinitionen können durch Mustervergleich (*pattern matching*) beschrieben werden.

Argumente werden erst dann ausgewertet, wenn sie gebraucht werden (verzögerte Auswertung, *lazy evaluation*).

Die aktuelle Version ist Haskell 2010.

Haskell beeinflusste u.a. Python und Scala und ist die Basis für funktional-logische Sprachen wie Curry.

Die 1990er

Java entstand –unter dem Namen Oak– 1991 in einem Team um James Gosling bei Sun Microsystems.

Man wollte eine plattformunabhängige, objektorientierte Software für Geräte der Unterhaltungselektronik entwickeln, die –z.B. über Fernsehkanäle verschickt– in Endgeräte geladen wird.
Zunächst war diese Entwicklung kein großer Erfolg.

Das World Wide Web (WWW) eröffnete neue Anwendungsmöglichkeiten für das System, das dann den Namen Java bekam.

Die 1990er

1996 wurde die Entwicklungsumgebung Java Development Kit (JDK) vorgestellt.

Java-Programme werden vom Compiler in eine Zwischensprache, den Java Byte Code, übersetzt. Dieser wird dann auf dem Zielrechner mit einem Interpretierer, der **Java Virtual Machine** (**JVM**), ausgeführt.

Java enthält viele Konzepte der objektorientierten Programmierung und basiert aus Gründen der besseren Akzeptanz auf C bzw. C++.
Allerdings verzichtet Java auf einige komplexere Elemente von C++, z.B. auf die Mehrfachbeerbung von Klassen.

Die 1990er: TPK-Algorithmus in Python

Python wurde Anfang der 1990er von Guido van Rossum als Lehrsprache entworfen.
Ziel ist darum ein gut lesbarer Quellcode.
Blöcke werden nicht geklammert, sondern durch Einrückung gekennzeichnet (wie bei Miranda und Haskell).

```
import math

def f(x):
    return math.sqrt(abs(x)) + 5 * x**3

vals = [float(raw_input()) for i in range(11)]
for i, x in enumerate(reversed(vals)):
    y = f(x)
    print('{0}: {1}'.format(i, y if y <= 400 else 'TOO LARGE'))
```

Die 2000er

Scala (*scalable language*) entstand 2001–2003 in der Schweiz um Martin Odersky.
Die Sprache ist objekt-orientiert und funktional.

Sie kann u.a. auf der Java Virtual Machine ausgeführt werden.

Jeder Wert in Scala ist ein Objekt; es gibt in der Sprache keine primitiven Datentypen mehr.
Statt einer Mehrfachbeerbung können Klassen explizit um (vorhandene) Implementierungen erweitert werden.

Die 2010er

Aktuell entwickeln große Konzerne eigene Programmiersprachen:

- Google die Sprachen **Go** (2009) und **Dart** (2011),
- Microsoft die Sprache **TypeScript** (2012) und
- Apple die Sprache **Swift** (2014).

Eine konzernunabhängige Entwicklung ist **Rust** (2012).

Überblick über Programmiersprachen

Einen Überblick über eine Vielzahl von Programmiersprachen und Hinweise auf weitere Informationen zu einzelnen Sprachen bekommt man über die WWW-Seiten von
Éric Lévénez (`http://www.levenez.com/lang/`) und
Bill Kinnersley
(`http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm`).