Eine kleine Geschichte der Programmiersprachen

Assembler, C, C++, C#, COBOL, Fortran, Java, Pascal, Visual Basic – das ist nur ein Teil der Programmiersprachen. Wir zeigen, warum es so viele verschiedene Sprachen gibt und worin sie sich unterscheiden.

Die Entwicklung der Programmiersprachen ist eng mit der Maschinensprache eines Computers verbunden. Wie der Name bereits sagt, enthält diese Sprache Befehle, die eine Maschine ausführen kann. Damit sind Mikroprozessoren gemeint, die in einem Computer verbaut sind (CPU, Festplattensteuerung etc.). Diese Prozessoren gehorchen speziellen Maschinenbefehlen. Sie werden zu einem binären Maschinenprogramm zusammengefasst, das der Computer Befehl für Befehl ausführen kann. Solche Programme sind für den Menschen schwer zu lesen und noch weit schwerer zu entwickeln.

Um diese Entwicklung zu erleichtern, hat man Programmiersprachen, Texteditoren und Übersetzungsprogramme erfunden. Die erste derartige Programmiersprache erschien im Jahr 1948 und nennt sich Assembler-Sprache. Der Programmierer schreibt ein Assembler-Programm in einem Texteditor und speichert es als textuelle Datei. Diese Datei kann der Computer noch nicht direkt ausführen. Dazu bedarf es eines Hilfsprogramms namens Assembler. Er übersetzt die komplette Datei, den sogenannten Quellcode, in ein Maschinen-programm (siehe folgende Abbildung). Erst dieses (binäre) Maschinenprogramm kann der Computer ausführen.

Der Assembler-Quellcode ist leichter zu verstehen als ein Maschinenprogramm. Er besteht aus einer Reihe kleinteiliger Befehle. Erst sehr viele dieser Befehle ergeben eine größere Programmfunktion. Schon ein einfaches Hallo-Welt-Programm ist daher deutlich länger als eines in einer Hochsprache wie Pascal (siehe folgende Abbildung). Ein großes Assembler-Programm zu schreiben, dauert sehr lange. Es ist zudem an einen bestimmten Prozessor gebunden und im Vergleich zu Hochsprachen wie Pascal schlechter lesbar. Diesen Nachteilen steht entgegen, dass Assembler-Programme

bei guter Programmierung oftmals schneller ablaufen und meist weniger Haupt- sowie Festplattenspeicher benötigen.

Trotz der unbestrittenen Vorteile der Assembler-Sprache waren deren Nachteile wie schlechte Lesbarkeit und Wartbarkeit, mangelhafte Entwicklerproduktivität und vor allem die Abhängigkeit von einer *bestimmten* Hardware (Mikroprozessor) so eklatant, dass sich schon ein paar Jahre später Sprachen der zweiten Generation entwickelten. Zu diesen ersten Hochsprachen gehörten Fortran und COBOL. Fortran wurde von der IBM ab 1954 mit dem Ziel entwickelt, eine leistungsstarke, hardwareunabhängige Sprache für naturwissenschaftliche Zwecke zu schaffen. Der Name ist Programm: Fortran ist die Kurzform von Formula Translation und beschreibt den Kerngedanken der Sprache schon sehr gut, die Übersetzung von Formeln.

Während Assembler-Programme mittels eines Assemblers zu einem Maschinenprogramm übersetzt werden, nutzt Fortran hierfür einen sogenannten Compiler. Der geänderte Name drückt schon aus, dass dieses Übersetzungsprogramm den Leistungsumfang eines Assemblers deutlich übersteigt. Ein Compiler übersetzt nicht nur den Quellcode einer Hochsprache in Maschinencode. Er optimiert hierbei das Programm auf maximale Geschwindigkeit und minimalen Speicherbedarf.

Mit anderen Worten: Der Compiler ist dazu ausgelegt, ein hocheffizientes Maschinenprogramm zu erzeugen. Warum war das so wichtig? Anfang der 1950er Jahre verfügten die damaligen Rechner im Vergleich zu den heutigen Computern nur über vergleichsweise leistungsschwache Prozessoren und extrem wenig Speicherplatz. Hätten die Compiler keine hocheffizienten Maschinenprogramme erzeugen können, hätten sich die ersten höheren Programmiersprachen vermutlich nicht so schnell durchgesetzt.

Vom Assembler zu den ersten Hochsprachen

Der Grundgedanke, der zu Fortran führte, hat Ende der 1950er Jahre auch zu der Programmiersprache COBOL inspiriert. Auch hier standen der Gedanke an Hardwareunabhängigkeit und die fachliche

Problemstellung im Vordergrund. Im Unterschied zu Fortran sollten damit aber keine naturwissenschaftlichen, sondern betriebswirtschaftliche Programme entwickelt werden, daher auch der Name COBOL.

Die Abkürzung steht für „Common Business Oriented Language“. COBOL lehnt sich stark an die natürliche Sprache an und ist im Vergleich zu Fortran auf die Verarbeitung großer Datenmengen ausgelegt. Die Sprache hat sich bald nach ihrer Einführung zu einer der meist genutzten Programmiersprachen entwickelt und ist heute immer noch weit verbreitet.

Die beiden Sprachen Fortran und COBOL wiesen anfangs eine Reihe von Defiziten auf, die erst nach und nach behoben werden konnten. Sie führten oftmals zu schlecht strukturierten und schwer zu pflegenden Programmen. Da die Anwendungen im Laufe der Jahre immer komplexer wurden, benötigten die Entwickler auch immer länger für die Programmierung, was zum Scheitern von diversen Projekten aufgrund der Überschreitung des Kostenrahmens führte.

Die steigende Zahl der gescheiterten Projekte löste Mitte der 1960er Jahre die erste Softwarekrise aus. Es gab verschiedene Ansätze, diese Krise zu überwinden. Neben verbesserten Entwicklungsprozessen und der Einführung bewährter Programmierbibliotheken entstanden auch weitere Programmiersprachen, die die Entwicklung kostengünstigerer Programme erlauben sollten.

Allen voran muss man hier die Programmiersprachen Pascal und C hervorheben. Pascal wurde von Niklaus Wirth 1971 auf der Grundlage der Programmiersprache Algol 68 entwickelt. Die Sprache wurde anfangs weniger für kommerzielle Anwendungen eingesetzt. Stattdessen verbreitete sie sich stark an Hochschulen, weil sie sich gut zum Erlernen der strukturierten Programmierung eignete.

Ein weiteres Plus von Pascal ist die strenge Typisierung. Das bedeutet, dass Variablen bereits bei der Übersetzung mittels Compiler einem festen Datentyp zuordnet sind, der nachträglich nicht geändert werden kann. Die Merkmale von Pascal führten dazu, dass die Programme sauberer aufgebaut werden konnten und Fehler schon im Ansatz

vermieden wurden. Das fördert die Entwicklung leicht wartbarer, robuster Programme.

Etwa zur selben Zeit wie Pascal entwickelte Dennis Ritchie an den Bell Labs die Programmiersprache mit dem minimalistischen Namen C. Sie basiert auf dem Vorläufer, der Programmiersprache B, daher ihr Name. C wurde zur besseren Programmierung des Betriebssystems Unix geschaffen und verbreitete sich entsprechend schnell in der Systemprogrammierung.

Da C eine universelle Programmiersprache ist, hat sich die Sprache aber auch für die Anwendungsentwicklung durchgesetzt. C-Programme sind entsprechend ihrem Einsatzzweck auf Portabilität und Effizienz getrimmt. Sie laufen dank der einfachen Syntax und der ausgereiften Compiler in der Regel sehr schnell ab. Die Kehrseite der Medaille sind einige sicherheitskritische Funktionen, die die Entwicklung leicht wartbarer, robuster Programme nicht gerade erleichtern.

Von den Hochsprachen zur OO-Programmierung

Auch fortgeschrittene Programmiersprachen wie Algol 68, Pascal und C waren nicht so perfekt, dass sie das Ende der Entwicklung von Programmiersprachen eingeläutet hätten. Anfang der 1970er Jahre kam die objektorientierte Programmierung (OOP) auf, die sich aber erst Mitte der 1980er Jahre durchsetzte. Der Grundgedanke der objektorientierten Programmierung besteht darin, die natürliche Welt besser als in den Vorgängersprachen C und Pascal in einem Computerprogramm abbilden zu können.

Neue Ansätze wie die Vererbung und erweiterte Konzepte wie die in Vorgängersprachen bereits vorhandene Kapselung und Polymorphie sollten helfen, die Programme übersichtlicher und robuster zu gestalten. Zu den einflussreichsten objektorientierten Sprachen zählen Simula-67, Smalltalk, Objective-C, C++, C#, Java und Visual Basic.

Während Simula-67 und Smalltalk heutzutage praktisch keine Rolle mehr spielen, haben die verschiedenen C-Derivate, allen voran C++, große Bedeutung erlangt. C++ ist eine Weiterentwicklung der Sprache

C und wurde im Jahr 1985 vorgestellt. Wie C ist diese Allzweck-programmiersprache mit dem Fokus auf die Systemprogrammierung entwickelt worden. Durch ihre universelle Verwendbarkeit verbreitete sie sich aber auch sehr schnell in der Anwendungsentwicklung.

C++ verwendet ebenfalls einen Compiler, ist hardwareunabhängig, aber viel schwerer als C zu erlernen, da neben den bestehenden Konzepten des Vorgängers C die neuen Sprachelemente der objektorientierten Programmierung hinzukommen. C++ stand oft in der Schusslinie von Kritikern, da die Sprache schwer zu erlernen ist und die Programme oft fehleranfällig sind. Selbst der Erfinder von C++, Bjarne Stroustrup, sagte einmal: „In C++ ist es schwierig, sich selbst in den Fuß zu schießen. Aber wenn man es tut, dann ist gleich das ganze Bein weg."

Das von Microsoft vorgestelle Visual Basic (VB) ist das glatte Gegenteil von C++. Hier steht die Einfachheit im Vordergrund. Statt Symbolen verwendet Visual Basic Wörter der englischen Sprache. Zudem setzte Visual Basic anfangs den leichter zu bedienenden Interpreter statt eines Compilers zur Programmübersetzung und -ausführung ein. Ein Interpreter übersetzt den Quellcode Instruktion für Instruktion in Maschinencode, so dass der Mikroprozessor ihn ausführen kann.

Bei neueren VB-Versionen für das .NET-Framework ist das Verfahren inzwischen geändert worden. Ähnlich der Programmiersprache Java kommt hier wieder ein Compiler zum Einsatz. Er übersetzt den VB-Quelltext in ein Zwischenformat, das eine sogenannte virtuelle Maschine ausführt. Laut RedMonk gehört Visual Basic nicht mehr zu den 20 populärsten Programmiersprachen. In dem neuesten RedMonk-Ranking sind beispielsweise Java und C# inzwischen deutlich beliebter.

Kommen wir also zu den derzeit angesagten Programmiersprachen Java und C#. Letztere wurde von Microsoft nach dem Java-Muster entwickelt und ist mehr oder weniger systemabhängig. Das ursprünglich von Sun Microsystems entwickelte Java wirbt hingegen damit, vollkommen systemunabhängig zu sein. Beide

Programmiersprachen sind mit C++ verwandt, aber vom Sprachumfang stark reduziert und daher leichter zu erlernen als C++.

Das 1995 vorgestellte Java verfügt beispielsweise im eigentlichen Sinne nicht über Zeiger und besitzt stattdessen eine automatische Speicherverwaltung. Wie C++ verwendet Java einen Compiler. Dieser erzeugt aus dem Java-Quellcode jedoch keinen Maschinencode für einen realen Mikroprozessor. Stattdessen erzeugt der Compiler sogenannten Bytecode, der von einer virtuellen Maschine (einem speziellen Computerprogramm) ausgeführt wird (siehe Abbildung oben). Da der Bytecode für jedes Betriebssystem identisch ist, laufen Java-Programme unter jedem Betriebssystem, auf dem eine passende virtuelle Maschine existiert.

Fazit

Auch mit Java und C# ist die Entwicklung der Programmiersprachen keineswegs abgeschlossen. Die Liste der Programmiersprachen könnte mit Groovy, Python, Ruby, Scala und vielen anderen neuen Sprachen nahezu beliebig lange fortgesetzt werden. Summa summarum lässt sich feststellen, dass es von der Maschinensprache zu den heutigen Programmiersprachen ein langer Weg war.

Die Programmiersprachen sind immer besser geworden, und die Unterschiede der neuen Programmiersprachen zu den Vorgängern fallen immer geringer aus. Ist ein Ende der Entwicklung der Programmiersprachen in Sicht? Das wohl nicht, aber die Entwicklung – das ist deutlich zu erkennen – beruhigt sich allmählich.