Einführung in die Programmiersprache BASIC (Teil 1)

Maximilian Herberger

Vorbemerkung

Einen BASIC-Programmierkurs in einer Zeitschrift wie IuR zu beginnen, bedarf in verschiedener Hinsicht der Rechtfertigung und Erläuterung. Der heutige erste Beitrag der geplanten Serie beschränkt sich im wesentlichen auf diese zur Einleitung notwendigen Hinweise und skizziert das vorgesehene Programm. Dabei sind allgemeine Überlegungen zu den Eigenschaften von Programmiersprachen nicht ganz zu umgehen, obwohl diese im ersten Teil einer Einführung notwendigerweise sehr generell gehalten sein müssen. Sie werden deshalb am Ende der Serie nochmals aufgegriffen und vertieft werden, da dann auf Grund der Kenntnis einer Programmiersprache eine detailliertere Behandlung möglich ist. Sollte das Leserinteresse ausreichend groß sein, könnten sich weitere vergleichbare Einführungskurse etwa zu PASCAL oder PROLOG anschließen.

1. Warum eine weitere Einführung zusätzlich zu den vielen bereits vorhandenen?

Es läßt sich nicht leugnen, daß es schon eine Vielzahl guter Einführungen in die Programmiersprache BASIC gibt. Daher stellt sich notwendigerweise die Frage, ob es wirklich unumgänglich ist, diese Literatur um einen weiteren Titel zu vermehren. Es gibt einen durchschlagenden Grund dafür, der mit der Situation im Bereich der Logik zu vergleichen ist. Auch dort fällt die Fülle ausgezeichneter Lehrbücher auf. Trotzdem steht der Jurist, der ein solches Lehrbuch durchgearbeitet hat, vor einem spezifischen Übertragungsproblem: Es ist keine triviale Aufgabe, die erlernten logischen Strukturen im juristischen Stoff zu erkennen. Das ist die Rechtfertigung dafür, den eingeführten Logiklehrbüchern Veröffentlichungen an die Seite zu stellen, die unter besonderer Berücksichtigung der juristischen Anwendungssituation in die Logik einführen. Im Bereich der Programmiersprachen ist die Lage ähnlich. Gewiß ist die Übertragung der erlernten Programmierkenntnisse in den juristischen Bereich dort vergleichsweise unproblematisch, wo es ausschließlich um vom Juristen auszuführende Berechnungen geht. Schwieriger wird es aber schon dort, wo sich beispielsweise die Frage stellt, wie man juristische Deduktionen in einer Programmiersprache abbilden kann. Das ist, wie das im Laufe des Kurses zu entwickelnde kleine Expertensystem zeigen wird, durchaus möglich. Nur ist der Weg von den erlernten Bausteinen der Programmiersprache zu dem vom Juristen praktisch benötigten Gebäude so weit, daß eine Hilfestellung dabei nicht als überflüssig erscheint.

2. Warum eine Einführung in die Programmiersprache BASIC?

Professionelle Programmierer sind oft nicht geneigt, BASIC als brauchbare Programmiersprache zu akzeptieren. Das gleiche gilt für einige Vertreter der Programmiertheorie, die BASIC in unterschiedlicher Schärfe kritisch beurteilen. Das oft kolportierte Diktum eines bekannten Informatikers, nach dem angeblich der für das weitere Programmiererleben verdorben sein soll, der seine Laufbahn mit BASIC begonnen hat, gibt diese Stimmungstendenz recht anschaulich wieder. Die im Rahmen dieser Kontroverse ausgetauschten Argumente können an dieser Stelle aus dem eingangs angedeuteten Grund nicht zureichend analysiert werden. Trotzdem soll andeutungsweise versucht werden, die Entscheidung für BASIC zu rechtfertigen, soweit die Zielsetzung dieser Einführung betroffen ist.

„BASIC" ist die Abkürzung für „Beginner's All Purpose Symbolic Instruction Code". Mit dieser Bezeichnung wurde ein doppelter Anspruch zum Ausdruck gebracht: Diese Programmiersprache sollte für Anfänger leicht zu erlernen sein und außerdem prinzipiell die Eignung haben, bei allen Problemstellungen anwendbar zu sein. Was die leichte Erlernbarkeit angeht, liegen mittlerweile vielfältige Erfahrungen vor. Schon mit Hilfe der einfachsten, in BASIC vorhandenen Sprachelemente, können Schüler nach wenigen Stunden Programme schreiben, die nicht mehr als völlig trivial zu bezeichnen sind. Was die Verwendbarkeit in vielfältigen Anwendungssituationen angeht, genügt ein Blick auf die enorme Masse der bisher in BASIC realisierten, durchaus anspruchsvollen Programme, um zu erkennen, daß der darauf bezogene Anspruch nicht völlig realitätsfern ist. Es gibt in BASIC geschriebene Betriebssysteme, Datenbankprogramme, Statistikpakete und sogar Expertensysteme. Zwar kann man einige spezielle Programmierprobleme beschreiben (z.B. bestimmte rekursive Fragestellungen), die in BASIC nicht ohne weiteres lösbar sind. Das ändert aber im Ganzen nichts an der angedeuteten universellen Brauchbarkeit.

Ein zugegebenermaßen gravierender Nachteil von BASIC (übrigens nicht von BASIC allein) ist darin zu sehen, daß es keinen allgemein anerkannten Standard

gibt, sondern zahlreiche BASIC-„Dialekte". Zwar ist die Übersetzung zwischen den verbreitetsten BASIC-Versionen nicht mit allzu großen Schwierigkeiten verbunden, zumal sich die Unterschiede oft nur auf wenige, im Programmablauf leicht zu identifizierende Punkte (wie etwa die Dateibehandlung) beschränken. Trotzdem ist die Situation unbefriedigend. Sie hat zur Konsequenz, daß man sich in einer Einführung zunächst auf einen BASIC-„Dialekt" beschränken muß. Für die vorliegende Serie wurde eine der populärsten Versionen ausgewählt, das von der Firma Microsoft entwickelte Microsoft-Basic. Der dabei zugrundegelegte Befehlsumfang ist auch in der auf dem IBM-PC vorhandenen BASIC-Version enthalten, so daß in dieser Reihe beschriebene Programme auf diesem (man denke an die vielen dazu kompatiblen Rechner) immer mehr zum „Industriestandard" werdenden Personalcomputer lauffähig sind.

Ein Vorteil von BASIC ist in seiner weiten Verbreitung und in seiner Kostengünstigkeit zu sehen. Wenn heute ein Personalcomputer gekauft wird, ist BASIC in aller Regel im normalen Lieferumfang enthalten. Handelt es sich um einen der IBM-PC's oder der dazu kompatiblen Rechner, so ist es ein BASIC mit dem hier verwandten Befehlsumfang. Obwohl auch andere Sprachen preiswert zu erwerben sind und sich schnell verbreiten (als besonders auffälliges Beispiel sei Turbo-Pascal von Borland-International genannt), dürfte BASIC in dieser Hinsicht doch immer noch der Spitzenreiter sein.

Ein weiterer Vorteil von BASIC besteht darin, daß es sich um eine „interpretierende" Sprache handelt. Darunter ist zu verstehen, daß jede Programmzeile während des Programmablaufs in eine von der Maschine ausführbare Version „übersetzt" wird (daher die Bezeichnung „interpretierend"). Dieses Prinzip hat zur Konsequenz, daß man einzelne Programmzeilen direkt ausführen kann. Besonders für den Anfänger, der erfahrungsgemäß viele Programmierfehler macht, schafft dies eine Arbeitsumgebung mit beachtlichen Vorzügen. Worin diese liegen, wird deutlich, wenn man einen kurzen Blick auf die „kompilierenden" Sprachen wirft. In diesen Sprachen wird das Programm (wie in BASIC) als Text geschrieben und dann (das ist der Unterschied zu einer „interpretierenden" Sprache) insgesamt in eine von der Maschine direkt ausführbare Version umkodiert. (Diesen Vorgang bezeichnet man als „Kompilieren".) Hat man nun beim Entwurf des Programms syntaktische Fehler gemacht, so werden diese beim Kompilieren erkannt und angezeigt. Man muß dann zu dem ursprünglichen Programmtext zurückgehen, diesen korrigieren, erneut kompilieren und das so lange, bis keine Syntaxfehler mehr vorhanden sind. Es ist leicht einzusehen, daß dieser Vorgang umständlicher ist als die Fehlerkorrektur in einer interpretierenden Sprache, so daß gerade für Anfänger BASIC leichter zu handhaben ist als kompilierende Sprachen. Allerdings kann nicht verschwiegen werden, daß interpretierende Programme wegen der bei Ablauf Schritt für Schritt notwendigen Übertragung in einen ausführbaren Maschinencode langsamer sind als kompilierte Programme, die diese Umsetzung bereits hinter sich haben. Aus diesem Grunde sind spezielle BASIC-Compiler entwickelt worden, mit denen man fertige und ausgetestete BASIC-Programme kompilieren kann. Auf diese Weise bleibt die vorteilhafte Entwicklungsumgebung einer interpretierenden Sprache erhalten, und man hat trotzdem anschließend die Möglichkeit, schnellen ausführbaren Code zu bekommen. Der Geschwindigkeitsnachteil schlägt also im Ergebnis nicht zu Lasten von BASIC zu Buche.

Zum Abschluß dieses Gedankenganges sollen noch einige der mehr programmiertechnischen Punkte angedeutet werden, die in der Debatte um BASIC und andere Programmiersprachen eine Rolle spielen. So ist es beispielsweise in BASIC möglich, außerordentlich schlecht strukturierte und unübersichtliche Programme zu schreiben. Dieses Ergebnis kann u.a. dann eintreten, wenn der (übrigens nicht nur in BASIC vorhandene) GO TO - Befehl unüberlegt verwendet wird. Dieser Befehl hat den Zweck, im Programm einen Sprung an eine bestimmte Stelle zu erlauben. Es ist leicht einzusehen, daß ein Programm nur sehr schwer nachzuvollziehen ist, wenn man auf diese Weise vorwärts und rückwärts springt. Nur ist darin kein Fehler der Programmiersprache zu sehen, sondern eine Folge der unzweckmäßigen Verwendung von zur Verfügung stehenden Programmierelementen. Achtet man von vornherein darauf, einen derartigen schlechten Programmierstil zu vermeiden, so sind auch in BASIC gut strukturierte und übersichtliche Programme möglich. Zwar stellen hier andere Programmiersprachen (wie etwa PASCAL) ausgereiftere Hilfsmittel und Kontrollstrukturen zur Verfügung, die in gewisser Weise den Programmierstil prägen und manches verwirrende Konstrukt ausschließen bzw. erschweren. Nur ist der Unterschied zu gut redigierten BASIC-Programmen im Regelfalle nicht so groß, daß dies BASIC für einen propädeutischen Kurs, der dem genannten Umstand Rechnung trägt, ausschließen würde.

Ein anderes, oft gegen BASIC ins Feld geführtes Argument ist der Hinweis auf die Tatsache, daß BASIC nur globale Variablen kennt. Darunter ist folgendes zu verstehen: Eine Variable in einem Programm ist ein Platzhalter für wechselnde Werte. Verwendet man etwa die Variable „Anzahl", um dieser Variablen die wechselnde Anzahl jeweils anwesender Personen zuzuweisen, so hat diese Variable den Wert 3, wenn drei Personen anwesend sind, den Wert 7, wenn sieben Personen anwesen sind etc. Programmtechnisch reserviert die Variable „Anzahl" einen Platz, an dem die jeweils wechselnden Werte abgespeichert werden. Ist diese Variable nun „global", so wird sie durch jede auf sie bezogene Operation an jeder beliebigen Stelle des Programms berührt. Darin liegt eine große Gefahr. Unter Umständen führt man eine Variable in der Meinung ein, sie sei „neu", während sie doch bereits an einer anderen Stelle des Programms vorkommt. Auf

Grund der Globalität der Variablen entstehen dann unerwartete und schwer zu eruierende Seiteneffekte und Nebenwirkungen. Andere Programmiersprachen (wie etwa PASCAL) kennen deshalb die Kontrollstruktur, daß eine Variable „lokal" verwandt, d.h. auf einen bestimmten Teil eines Programms beschränkt werden kann. Dann sind Effekte der eben beschriebenen Art ausgeschlossen, was sicher ein Vorzug ist. Nur muß man bei der Gesamtbilanz hier (wie auch beim vorherigen Punkt) berücksichtigen, daß die Probleme in BASIC auf Grund einer nicht optimalen Verwendung der Sprache entstehen. Wer die Verwendung der Variablen in BASIC gut plant, kann beispielsweise die Variablenbezeichnungen so abwechseln, daß im Ergebnis bestimmte Variablen nur lokal vorkommen. Die bisherige Skizze konnte keine Gesamtbilanz sein. Sie ist aber vielleicht ausreichend, um folgende Annahme plausibel zu machen: BASIC hat genügend Vorteile, um für das Erlernen einer Programmiersprache in Frage zu kommen. Die Nachteile können ausgeglichen werden, wenn man bei der Vermittlung bestimmte notwendige Elemente eines klaren und übersichtlichen Programmierstils ausreichend betont. Außerdem (und dies wird eines der abschließenden Themen des Kurses sein) gibt es mittlerweile neue BASIC-Versionen, die viele der an BASIC kritisierten Nachteile ausgleichen und es ermöglichen, unter dem hier zugrundegelegten Standard geschriebene Programme zu übernehmen. Bei der Verwendung einer derartigen BASIC-Version sieht die Bilanz noch günstiger als hier gezeichnet aus.

3. An wen richtet sich die Serie?

Festzuhalten ist zunächst, daß der Kurs sich an Leser wendet, die keine Vorkenntnisse im Programmieren haben. Der Leser mit Vorkenntnissen, der trotzdem diese Beiträge liest, sollte sich also nicht an dem elementaren Charakter der Ausführungen stören.

Mit der genannten Maßgabe richtet sich die Einführung zunächst (deshalb auch die Plazierung in der Rubrik „EDV und Jura-Ausbildung") an den Jura-Studenten, der überzeugt ist, daß für seine künftige Berufspraxis Programmierkenntnisse von Nutzen sind. Die entsprechende Prognose ist übrigens bereits unter den gegenwärtigen Umständen ausreichend erhärtet. Die zunehmende Anzahl von Juristen wird in immer stärkerem Maße dazu führen, daß bei der Auswahl für Berufspositionen Zusatzkenntnisse eine entscheidende differenzierende Rolle spielen. Alles spricht dafür, daß fachbezogene Programmierkenntnisse dabei nicht an letzter Stelle ins Gewicht fallen. Es ist deswegen kein Zufall, daß diese Serie auch auf Anregungen von Studenten zurückgeht, die ein entsprechendes Ausbildungsangebot vermissen. Sie stützt sich im übrigen ebenfalls auf Erfahrungen, die im Rahmen der Programmierarbeit mit Studenten gemacht wurden.

Adressat der Serie könnte aber auch der schon im Beruf stehende Jurist sein. Um zu zeigen, daß diese Vorstellung nicht völlig an der sich entwickelnden Berufswirklichkeit vorbeigeht, sei beispielsweise darauf hingewiesen, daß die American Bar Association in dem von ihrer technischen Arbeitsgruppe herausgegebenen „Jurimetrics Journal" einen von Ashley S. Lipson konzipierten „Lawyer's Short Course in Basic Computer Language" veröffentlicht hat (vol. 24, 1984, S. 154 - 163). Der vorliegende Kurs ist zwar ausführlicher konzipiert als der von Lipson, teilt aber die Grundüberzeugung dieses Autors, daß ein Anwalt ohne gewisse Grundkenntnisse im Bereich der Programmiersprachen tendenziell nicht mehr über die nötige umfassende Beurteilungskompetenz verfügen wird, die er an seinem Arbeitsplatz benötigt. Ein weiterer Hinweis darauf, wie nützlich Programmierkenntnisse für praktisch tätige Juristen sein können, ist in der steigenden Anzahl von Programmen zu sehen, die solche Juristen für ihren eigenen Bedarf schreiben. Ein Beispiel dafür ist das in den ersten drei Heften von IuR vorgestellte Programmpaket von Friederici, das in diesem Programmierkurs durchgehend als ein Veranschaulichungsbeispiel herangezogen werden wird. Um noch zwei französische Erfahrungen zu zitieren: Die Komplexität des französischen Staatsangehörigkeitsrechts hat Juristen veranlaßt, ein Expertensystem zu programmieren (in BASIC übrigens), das die Feststellung der Staatsangehörigkeit zum Gegenstand hat. Außerdem existiert ein (ebenfalls in BASIC geschriebenes) Programm, das die Vorbereitung eines Revisionsschriftsatzes unter Einschluß einer Datenbankkomponente unterstützt. Und zwei der amerikanischen Telekommunikationsnetze bieten von Anwälten erstellte BASIC-Programme für ihre juristisch interessierten Benutzer an. Das sind sicherlich nur partielle Eindrücke, die aber doch, obwohl fragmentarisch, die Annahme unterstützen, daß Programmieren in immer selbstverständlicherer Weise in die juristische Berufspraxis integriert werden wird.

Schließlich könnte auch der Jurist an der Serie interessiert sein, der zwar nicht unbedingt selber programmieren will, wohl aber in seiner Berufspraxis mit der juristischen Beurteilung von Software in den verschiedenen Rechtsgebieten befaßt ist. Eine kompetente rechtliche Behandlung dieser Materie setzt Grundvorstellungen von der Eigenart des Produkts „Software" voraus. Glücklicherweise festigt sich immer mehr die Überzeugung, daß man beispielsweise schlecht über die Originalitätskriterien für Software sprechen kann, ohne wenigstens in Umrissen über eigene Eindrücke von Softwareerstellung zu verfügen. Gleiches dürfte etwa gelten, wenn die Mängelhaftung im Softwarebereich zur Debatte steht. Um auch diesem Interesse Rechnung zu tragen, wird der Kurs Programmierbeispiele integrieren, die speziell auf die Erläuterung derartiger Rechtsprobleme zugeschnitten sind und als Anwendungsbeispiele dienen können.

4. Wie sollte die Serie durchgearbeitet werden?

Die ersten Folgen des Kurses sind so aufgebaut, daß man sie ohne Einsatz eines Computers nachvollziehen

kann, obwohl erfahrungsgemäß schon hier der Einsatz eines Rechners die Mitarbeit sehr erleichert. Das entsprechende Darstellungsprinzip, das eine größere Ausführlichkeit erfordert, wird auch im weiteren Verlauf der Einführung beibehalten. Trotzdem dürfte es ab einem bestimmten Punkt sehr schwierig wenn nicht unmöglich werden, die komplexen Programme ohne Rechnereinsatz nachzuvollziehen. Es wird deswegen im Laufe der Reihe in einem gesonderten Beitrag ein preisgünstiger, auch für Studenten erschwinglicher Computertyp vorgestellt werden, auf dem alle im Laufe der Serie entwickelten Programme lauffähig sind. Dieser Computer soll dann gleichzeitig der Prototyp sein, auf den sich weitere IuR-Serien etwa zum computerunterstützten juristischen Lernen beziehen. Nach der Vorstellung dieses Computers werden in IuR entwikkelte Programme, wenn die Leser dies wünschen und die Autoren zustimmen, auch auf Diskette zur Verfügung stehen.

5. Welche Programme werden diskutiert oder entwickelt?

Nach der Vorstellung erster elementarer Bestandteile der Programmiersprache BASIC, wird (wie oben bereits erwähnt) das in den drei ersten IuR-Heften vorgestellte Programm von Friederici näher erläutert werden. Dadurch sollen auch die Leser, die den programmiertechnischen Teil des Beitrags nicht oder nicht vollständig nachvollziehen konnten, in den Stand versetzt werden, Konstruktion und Ablauf dieses Programms im Detail zu beurteilen und gegebenenfalls ähnliche Vorhaben selbständig zu realisieren.

Es zeichnet sich ab, daß auch in ansonsten nicht EDV-orientierten juristischen Fachzeitschriften Aufsätze zu lesen sind, die unter Verwendung von BASIC-Programmbeispielen Fragen aus dem Berührungsbereich von Informatik und Recht erörtern. Ein besonders interessantes Beispiel dafür ist beispielsweise der Beitrag von Schreiber zum Thema „Rechtsanwendung durch Computer" (ohne Fragezeichen !) in Jura 1985 (S. 288 - 290). Abgesehen von der eher nebensächlichen Tatsache, daß der Druckfehlerteufel in dem Abdruck des BASIC-Programms einige kleinere Streiche gespielt hat (was immer passieren kann, wenn Programmlistings neu gesetzt werden), verdient die Tatsache Beachtung, daß hier unter Rückgriff auf das Programmierbeispiel folgende Annahme verfochten wird: „Die Verwendung von Computern in der Rechtsanwendung verspricht nicht nur die Lösung vieler Probleme in der Rechtswissenschaft. Auch ergibt sich hier eine neue Perspektive im Umgang mit rechtswissenschaftlichen Problemen" (a.a.O., S. 290). Diese bemerkenswerte These dürfte einer genaueren Betrachtung wert sein, was allerdings gründlich nur möglich ist, wenn man auch das von Schreiber angeführte Programmbeispiel analysiert, auf das sich die These ja stützen soll.

Ein weiteres Beispiel wird die Entwicklung eines kleinen Programms zur Fristenkontrolle und Überwachung in der anwaltlichen Praxis sein. Wie jeder Anwalt weiß, sind die dabei zu lösenden Probleme durchaus komplex. So kann etwa ein derartiges Programm Plausibilitätskontrollen bei Fristenumnotierungen integrieren, die helfen können, bestimmte typische Haftungsfälle zu verhüten. An diesem Programm wird man auch besonders deutlich sehen können, wie anwaltliche Berufserfahrung und von Programmierkenntnissen geleitete Designvorstellungen beim Entwurf eines praxisnahen Programms zusammenwirken müssen, will man möglichst gute Ergebnisse erzielen.

Eine der komplexesten Anwendungen im Rahmen dieses Programmierkurses wird der Entwurf eines begrenzten Expertensystems sein. Die dafür benötigten Komponenten werden im Laufe der Serie schrittweise vorgestellt und dann am Ende zu dem komplexeren Gesamtsystem zusammengefügt. Da sich die Regel- und Deduktionskomponente eines derartigen Systems nicht ohne Kenntnisse der juristischen Logik verstehen läßt, ist dann auch der Punkt gekommen, an dem das notwendige Zusammenspiel von juristischer Logik und Informatik in Ansätzen erläutert werden kann.

Oben wurde bereits erwähnt, daß auch auf besondere juristische Problemstellungen (Fehlerbegriff, Originalitätskriterium, zugesicherte Eigenschaft etc.) zugeschnittene Programmbeispiele in die Serie integriert sind. Damit berührt es sich, daß auch die Vorstellung und kritische Rezension derjenigen juristischen Programme vorgesehen ist, die im Quellcode vorliegen und in BASIC geschrieben sind. Teilweise finden sich dort nämlich Unstimmigkeiten (z.B. im Bereich der Rechengenauigkeit), die den juristischen Benutzer durchaus in die Irre führen können. Um nur ein Beispiel zu nennen, das im Bereich der mit BASIC programmierbaren Taschencomputer eine Rolle spielt: Die Wurzel aus einer Zahl zu ziehen und dann anschließend die Wurzel zu quadrieren erbringt nicht immer (wie es mathematisch sein müßte) die Ausgangszahl als Ergebnis. Auch Rundungsungenauigkeiten können eine fatale Rolle spielen. Es ist zu hoffen, daß diese notwendigerweise kurze und weitgehend theoretische Einführung in die neue Serie trotzdem die Konturen des Projekts ausreichend sichtbar machen konnte. Dadurch sollte gleichzeitig auch erreicht werden, daß alle an dem Kurs Interessierten noch ihre Anregungen beitragen können. Denn das vorgestellte Konzept bildet lediglich einen Rahmen.

(wird fortgesetzt)