Flußdiagramme — eine Brücke zwischen Programmierern und Programmiersprachen?

Wer hat sich als Programmierer nicht schon darüber geärgert, daß in einer Zeitschrift ein Listing eines interessanten Programms abgedruckt ist, aber ausgerechnet für einen anderen Computertyp als den eigenen? Flußdiagramme helfen, eventuelle Anpassungsprobleme auf ein Minimum zu beschränken.

Vielen Computerfreaks wird es wohl schon so ergangen sein. Man sieht in einer Zeitschrift ein interessantes Programm, aber es wurde für einen anderen als den eigenen Computer geschrieben. Also versucht man sein Glück und fängt an das Programm auf den eigenen Computer anzupassen. Meistens reicht dann die Beschreibung nicht aus, um in Verbindung mit dem Listing die nötigen Änderungen vorzunehmen. Nach einigen Stunden mühsamen Eintippens stellt man dann bedrückt fest, Basic ist nicht gleich Basic und das für den Computer XY geschriebene Programm will einfach nicht auf dem eigenen Gerät laufen. Entweder kapituliert man dann, oder man versucht solange weiter bis ein lauffähiges Programm entsteht, welches dann unter Umständen kaum noch Ähnlichkeit mit dem Original hat. Jedenfalls erreicht man dann irgendwann einmal den Punkt, wo man sich fragt, warum nicht alle Programme in einer einheitlichen Programmiersprache geschrieben werden, die von allen Computern verstanden werden können. Es ist natürlich nur mit großem Aufwand möglich, eine einheitliche Programmiersprache für alle bestehenden Computersysteme zu schreiben und außerdem wäre der Aufwand höher als der effektive Nutzen.

Es stellt sich also die Frage, ob es nicht eine Möglichkeit gibt, Programme so darzustellen, daß sie von der Hardware und der Programmiersprache weitgehend unabhängig sind. Hier stellen Flußdiagramme eine große Hilfe dar.

In der Datenverarbeitung kennt man schon lange verschiedene Möglichkeiten der Programmdarstellung. Einen besonderen Platz nimmt dabei der Ablaufplan ein. Er bietet mit seinen Sinnbildern (siehe Bild 1) die Möglichkeit ein Programm übersichtlich und detailliert darzustellen, ohne an einen bestimmten Computer gebunden zu sein.

Wie man von einer Problemstellung zu einer vollständigen Programmdokumentation kommt, soll nun an einem kurzen Beispiel erläutert werden.

Problem: Für Zahlen, die größer als 0 und kleiner oder gleich 100 sind, sollen die Quadratwurzeln berechnet werden. Um das Problem zu lösen, sollte für die einzelnen benötigten Programmteile ein Schema erstellt werden:

1. Eingabe der Zahl x
2. Überprüfung der Zahl x
3. Berechnung der Wurzel
4. Ausgabe des Ergebnisses

Ein Beispiel, das aus technischen Gründen auf Seite 15 abgedruckt werden mußte, zeigt wie das obige Schema als Programmablaufplan aussehen kann.

Anhand des ausführlichen Flußdiagramms sollte es den meisten Programmierern möglich sein, ein lauffähiges Programm zu schreiben. Die eben erwähnte Methode ist jedoch nicht die einzigste Möglichkeit einen Programmablaufplan zu dem gestellten Problem zu erstellen. In Anlehnung an die Programmiersprache Basic wurde hier die Beschriftung der Symbole geändert. Für diejenigen, die sich mit Basic auskennen, dürfte die gezeigte Version eine wesentliche Vereinfachung darstellen.

Das Beispiel zeigt, daß mehrere Möglichkeiten zur Verfügung stehen einen Programmablauf zu erstellen. Wie umfangreich die Diagramme ausfallen, hängt in erster Linie davon ab, ob man nur einen groben Überblick über das Programm vermitteln möchte, oder ob es anderen Programmierern die Möglichkeit geben soll, ein lauffähiges Programm zu erstellen.

Ein Programmablaufplan ist also, wenn genügend Informationen darin enthalten sind, eine nützliche Brücke zwischen Programmierern und Programmen. Wie stabil sie ist, hängt jedoch von einigen weiteren Faktoren ab. Eine möglichst ausführliche Programmdokumentation sollte noch Informationen enthalten, die benötigt werden, um entscheiden zu können, ob es überhaupt sinnvoll ist, das Programm auf das eigene System anzupassen.

(Burkhard Appe/rg)