In der Computerbranche gibt es wenige Dinge, die interessanter sind als das Entwerfen einer neuen Programmiersprache. Eine davon ist ADA - das Superspielzeug des Verteidigungsdepartements der Vereinigten Staaten. Wie Sie vielleicht wissen, wurde ADA entworfen, um veraltete und nicht mehr brauchbare Programmiersprachen (wie z.b. FORTRAN oder COBOL) zu ersetzen.

Die Schwierigkeit liegt darin, dass die Entwicklung einer Programmiersprache 20 bis 30 Jahre dauert und erst dann beginnt, wenn man wirklich überzeugt ist, dass die zur Verfügung stehenden Sprachen unbrauchbar sind. Dieser Zyklus kann umgangen werden, indem man direkt eine Sprache plant, welche ADA ersetzt bevor ADA überhaupt zum Einsatz kommt. Wenn dann nämlich ADA unbrauchbar geworden ist, steht schon die neue Sprache bereits zur Verfügung.

Die neue Generation der Entwerfer von Programmiersprachen haben die Gewohnheit angenommen, ihre geistigen Kinder nach lebenden oder gestorbenen Personen zu benennen, anstelle zu den üblichen, nichtssagenden Abkürzungen Zuflucht zu nehmen. Pascal wurde nach Blaise Pascal, dem Erbauer der ersten Rechenmaschine benannt. ADA ist der Vorname der ersten Programmiererin. Für unseren Namen stand Charles Babbage Pate, welcher in Armut starb, während er versuchte den ersten Computer zu bauen. Die neue Sprache ist also nach dem ersten Computerbauer benannt, welcher Budget und Zeitplan überschritt.

Babbage baut auf den Sprachelementen auf, welche entdeckt wurden, nachdem der Entwurf von ADA abgeschlossen war. So spricht C.A.R. Hoare in seiner Antrittsrede vor dem ACM in 1980 von zwei Möglichkeiten Software zu entwerfen: "Eine Möglichkeit besteht darin, sie so einfach zu machen, dass sie offensichtlich keine Nachteile hat. Die andere Möglichkeit ist, sie so komplex zu machen, dass sie keine offensichtliche Nachteile hat." Die Väter von Babbage haben sich entschlossen, die dritte Möglichkeit zu wählen, nämlich eine Sprache zu entwerfen, welche offensichtlich nur Nachteile hat. Programme, welche in Babbage geschrieben werden, sind so unzuverlässig, dass man mit dem Unterhalt beginnen kann, bevor sie überhaupt ins System eingebaut werden. Dies bedeutet für die Zukunft ein stetig steigender Bedarf an qualifiziertem Personal auf dem Datenverarbeitungssektor.

Babbage verlangt ebenso wie Pascal `strong typing' (starkes Eintippen), um Fehler bei der Verwendung von Daten unterschiedlichen Typs zu vermeiden. Die Väter von Babbage plädieren für `good typing', um in einem Programm Fehler der Rechtschreibung zu verhindern. Spätere Versionen von Babbage werden auch `touch typing' einschliessen und erfüllen damit einen lang gehegten Wunsch.

Ein äusserst heisses Eisen unter den Prüfsteinen einer Sprache ist die Parameterübergabe beim Aufruf von Unterprogrammen. Die einen sind für `call by name' (Aufruf mit dem Namen), andere für `call by value' (Aufruf mit dem Wert. Die Väter von Babbage stehen für `call by telephone' ein. Dies ist vor allem bei der Parameterübergabe über grössere Distanzen sehr effizient.

ADA legt alles Gewicht in die Portabilität der Software. Babbage verlegt sich mehr auf die Portabilität von Hardware. Was nützt ein Computer, wenn Sie ihn nicht mitnehmen können.

Es ist immer gut, wenn eine staatliche Stelle eine Sprache unterstützt. Bei COBOL war es die Verwaltung, bei ADA das Verteidigungsministerium. Nach eini-

gen Absagen konnte die Abteilung für Sanität als Sponsor von Babbage verpflichtet werden.

ADA erlaubt keine Teilsysteme. Babbage ist genau das Gegenteil. Unter Babbage ist nichts festgelegt, ausser seiner Ausbaubarkeit - jeder Benützer muss seine eigenen Version definieren. Babbage erlaubt jedem Benützer sein `Babbage' so gross oder so klein zu halten, wie er dies wünscht. Damit wird auch die endlose Diskussion, ob Babbage eine umfangreiche oder eine einfache Sprache sei, überflüssig. Babbage ist die ideale Sprache für die `Ich' Generation. Die folgenden Beispiele geben Ihnen einen guten Überblick über die Möglichkeiten von Babbage.

Strukturierte Sprachen haben das `goto' und die bedingte Sprungliste mit Bann belegt und durch das einfachere IF-THEN-ELSE ersetzt. Babbage kennt eine Vielzahl von neuen bedingten Anweisungen, welche wie Termiten in den Strukturen der Programme arbeiten werden.

@description{WHAT IF}@
Springt, bevor die Bedingung überprüft wird.

@OR ELSE@
Bedingte Drohung, wie z. B. Addiere zwei Zahlen OR ELSE!

@WHY NOT@
Führt den folgenden Kode aus nach dem Motto: Nach uns die Sintflut!

@WHO ELSE@
Wird vor allem beim Abfragen von verschiedenen Eingangslinien verwendet.

@ELSEWHERE@
Wo Ihr Programm wirklich ist, wenn man annimmt, dass es hier ist.

@GOING GOING GONE@
um unstrukturierte Programme zu schreiben. Springt in Ihrem Programm zufälligerweise irgendwohin. Erfüllt die Aufgabe von mindestens 10 `goto's.}

Während Jahren haben Programmierer `for', `do until', `do while' benützt um eine Schleife zu programmieren. Um diese Entwicklung zu unterstützen, offeriert Babbage folgende Konstrukte:

@description{DON'T DO WHILE NOT}@
Diese Schleife wird nicht ausgeführt, wenn die Bedingung nicht erfüllt ist (ausser das Programm wird an einem Freitag ausgeführt).

@DIDN'T DO@
Das Programm führt die Schleife einmal aus und verwischt dann jegliche Spur.

@CAN'T DO@
Die Schleife wird ausgepfiffen.

@WON'T DO@
Die Zentraleinheit stoppt, weil sie den Kode innerhalb der Schleife nicht mag. Die Ausführung kann wieder aufgenommen werden, wenn man am Terminal "May I" (darf ich) eingibt.

@MIGHT DO@
Hängt davon ab, wie sich die Zentraleinheit gerade fühlt. Die Schleife wird ausge-

führt, wenn sie sich in einem `Hoch' befindet, bzw. nicht ausgeführt, wenn sie sich in einem `Tief' befindet oder ihre Gefühle verletzt wurden..

@DO UNTO OTHERS@
Wird vor allem in Mehrplatzsystemen verwendet, um die Benützer gleichmässig zur Weissglut zu bringen.

@DO WAH@
Wird in den Taktgebern für Computer-Musik benützt (Rag timing).}

Jede strukturierte Sprache, die noch einigen Respekt vor sich selber hat, hat eine Fall - Anweisung (case), um Mehrfachsprünge programmieren zu können. ALGOL bietet eine indizierte case - Anweisung an und Pascal kennt eine bezeichnete case - Anweisung. Da bleibt keine grosse Auswahl. Babbage bietet eine reiche Auswahl von solchen Anweisungen:

@description{JUST IN case}@
um nachträgliche Einfälle und gefälscht Faktoren zu behandeln. Damit können Sie mit Null multiplizieren, falls Sie versehentlich mit Null dividiert haben.

@BRIEF CASE@
um portable Software zu unterstützen.

@OPEN AND SHUT case@
braucht keinen Richtigkeitsnachweiss.

@IN ANY case@
wird immer ausgeführt.

@HOPELESS case@
wird nie ausgeführt.

@BASKET case@
ein wirklich hoffnungsloser Fall.}

Das Babbage Entwicklungsteam untersucht andauernd neue Elemente, um die Benützer daran zu hindern, je einen Grad von Kenntnis und Effektivität zu erreichen. Beispielsweise sind die Entwickler von Babbage gerade daran zu überprüfen, ob ein `beinahe Gleichheitszeichen' beim Vergleich zweier Gleitkommazahlen sinnvoll sei oder nicht. Dieser Operator würde die Frage: Ist man nahe daran? ein für alle Mal lösen.

Keine Sprache, und sei sie noch so schlecht, kann alleine stehen. Für Babbage braucht es ein Betriebssystem, welches sämtliche Erkenntnisse auf diesem Gebiet mit einschliesst. Nach längerer Untersuchung entschlossen sich die Entwickler von Babbage für ein `virtuelles' Betriebssystem. Beinahe jeder hat ein Betriebssystem mit `virtuellem Speicher'. Das Team entschied sich etwas leicht anderes zu tun. Das neue Betriebssystem heisst VTOL, ein Abkürzung für `Virtual Time Operating System'. Während Anlagen mit virtuellem Speicher (d.h. Speicher, den man eigentlich gar nicht hat) arbeiten, macht VTOS das gleiche mit der Rechenzeit der Zentraleinheit.

Das führt zu dem Resultat, dass VTOS praktisch unendlich viele Programme zur gleichen Zeit ausführen kann. Die Anlagen mit virtuellem Speicher halten immer einen Teil der Information auf dem Plattenspeicher. Das Gleiche kann VTOS nur

mit einem Trick erreichen. Obschon scheinbar alle Programme gerade jetzt ausgeführt werden, laufen in Wirklichkeit einige erst nächste Woche.

Wie Sie sehen können steckt Babbage immer noch in den Kinderschuhen. Das Entwicklungs-Team von Babbage sucht weitere Vorschläge für diese wunderschönen, neue Sprache und als einziger Mitarbeiter dieses Teams (alle Anfragen um Mitgliedschaft werden angenommen) richte ich einen inständigen Appell an die ganze Datenverarbeitungs-Gemeinde:

Helfen Sie, diesen Traum Wirklichkeit werden zu lassen!