D

DATA PROCESSING

Before the microcomputer was developed, all computing was known as *data processing*. At that time, the high cost of mainframe and minicomputers limited their use to very large firms and public bodies. The hardware needed special air-conditioned environments, and required trained staff to operate and maintain it. The computers were invariably controlled by a Data Processing (or DP) department, and no one could gain access to the machine without the approval of the Data Processing Manager.

The microcomputer revolution meant an increase in accessibility. The micro, freed from the constraints imposed upon mainframes and minicomputers, could sit on an executive's desk, where it would be constantly available to perform whatever application was required. Data Processing departments still exist in large companies, and perform a wide variety of tasks. These applications include financial modelling, stock control, cost and management accounting, production planning, payroll and personnel records.

DEBUGGING

In its broadest sense, *debugging* is simply the correction of errors — whether these errors are mistakes in a computer program or malfunctions in the hardware. In particular, though, the term is used to refer to the systematic testing and correction of a computer program.

The person credited with coining the term is Captain Grace Hopper, who worked on the electromechanical Harvard Mark II computer in 1945. The cause of a particularly elusive error in a program was eventually traced to a moth that had been hammered to death in one of the machine's relay switches. When asked by a superior why progress was slow, Captain Hopper explained that she was 'de-bugging the machine'. From this derives the word *bug*, which is simply an error in a program's operation.

DECISION TABLE

A computer's usefulness is derived from its ability to act on conditional statements. The simplest of these is the BASIC IF . . . THEN statement, in which the computer follows a set course of action if a particular condition is met. All forms of computer decisions can ultimately be reduced to this form.

A *decision table* simply indicates the actions to be taken under various conditions. It is set up in two columns — one for the *conditions* and the other for the *actions* to be taken. The computer then works through the first column until it finds the entry that matches the required condition and takes the action indicated by the second column.

Implementing a decision table in BASIC by any means other than a collection of IF . . . THEN statements would be a very involved process, so machine code is generally used. The ON . . . GOTO structure is, however, a simple form of decision table, in which the list of conditions consists of a list of numbers, and the possible outcomes are all GOTO statements.

DECISION TREE

Like a decision table, a *decision tree* is a collection of conditions and specified actions. The difference between the two is that in a decision tree the computer does not need to search the entire list. Instead, it searches an initial section of the list and the resulting action statement then indicates the next group of conditions to be searched. The effect is like a tree — the program starts at the trunk, and each set of conditions that is met represents a split into several branches.

DECLARATION STATEMENT

The BASIC language requires all array variables to be DIMensioned before use. Some other programming languages insist on *all* variables being 'reserved' in this way, with the user having to specify whether they are integer, real, string or double-precision variables. The program lines that reserve variables (usually at the start of a program) are called *declaration statements.*

There is no fundamental reason why arrays must be declared, except for the fact that this makes the system more efficient. When a DIM statement is encountered, the BASIC interpreter sets aside an area of RAM proportional to the size of the array. Ordinary variables are then stored in RAM adjacent to the array variables. If a DIM statement was not used, each time the user addressed the array using a higher index than before, the operating system would need to move all the other variables in memory to make sufficient space.

It is good programming practice to simulate the declaration of all variables at the start of a program, in the order of frequency of use. Thus, if your program features many loops that use I as an index, the statement LET I = 0 in the first line of the program will make it run faster by ensuring that I is placed in the first position in the table of variables.