THE UNIVERSITY OF NOTTINGHAM

FACULTY OF APPLIED SCIENCE

Applications of Computers

LECTURE 8

"PROGRAMMING STRATEGY"

by

S. GILL, M.A., Ph.D.

"PROGRAMMING STRATEGY"

SUBROUTINES

Fundamentally, a subroutine is simply part of a programme. The reasons for considering a programme as made up of subroutines are three-fold:

1. By splitting the calculation into parts and specifying each part separately, the programmer is enabled to concentrate his attention on one thing at a time.

2. Each subroutine can be tested independently if desired by building it into a simple test programme.

3. It may happen, particularly in scientific calculations, that several of the subroutines have already been required in previous calculations, so that the same bit of programming can be used again.

For scientific work the last of these reasons is very important. Every new type of computer rapidly accumulates a collection of standard subroutines, known as a "library". Each subroutine in the library is covered by a specification showing exactly what calculation the subroutine performs and what conditions must be observed when it is used.

A number of different techniques have been worked out for incorporating subroutines into programmes. For example, there is a distinction between the "open" and "closed" form of subroutine. The former type is inserted amongst the other instructions of a programme, whereas the latter type stands on its own in the store and waits for control to be transferred to it by means of a jump instruction when it is required to operate. A closed subroutine is arranged to transfer control back to the rest of the programme when its operation is complete.

In order to make a library subroutine more generally useful, it is often arranged to accept parameters defining certain details of the operation to be performed. These parameters have been classified as "pre-set" and "programme" parameters. Pre-set parameters remain fixed in value throughout the entire calculation, whereas programme parameters are re-set every time the subroutine is used. Thus a subroutine with programme parameters may be used to perform a number of slightly different tasks during the same calculation, whereas a subroutine having only pre-set parameters always operates in exactly the same way during any one calculation.

One of the commonest types of subroutine is that used for the input or output of numbers to and from the computer. Numbers, as they are presented to the machine and as they are required to be printed out, exist in a variety of different forms, distinguished by different numbers of digits, different positions of the decimal point (if any), different disposition of spaces, etc. The computer designer cannot easily cater for the interpretation of all these forms automatically, and he therefore usually provides a rudimentary input or output instruction and leaves the programmer to construct subroutines for carrying out the necessary conversions between the internal form of numbers and the various forms which they take in the outside world. These subroutines are, of course, just as important in business as they are in scientific work. They make considerable use of parameters to define the exact form of the numbers being handled.

INTERPRETIVE ROUTINES

An interpretive routine has been described as a generalisation of the idea of a subroutine with programme parameters. In this case, however, the subroutine is capable of a variety of operations, and is capable of

accepting a long string of programme parameters. It works its way through these parameters one by one, carrying out an operation defined by each one. In fact the subroutine behaves towards these parameters in a similar manner to that in which the computer itself behaves towards its instructions. The parameters are in fact sometimes referred to as "interpretive instructions", and the programmer writes a "programme" of interpretive instructions in much the same manner as he writes an ordinary programme. The code by which they are interpreted is, however, not the instruction code of the machine itself, but a code determined by the interpretive routine. The possession of an interpretive routine thus gives the programmer the feeling of possessing a new type of computer, operating according to a different instruction code.

This is therefore an extremely powerful technique in programming, and is frequently used for the programming of lengthy calculations on elements which are not ordinary numbers, but which may be numbers expressed in a different form (e. g. floating point or double-length numbers), or even entirely different mathematical entities such as vectors, matrices, or group elements.

MISTAKES IN PROGRAMMES

A mistake in a programme often has the same effect as a fault in the computer, and it is sometimes hard to distinguish. The symptoms are often puzzling. The machine may print rubbish, it may enter a closed loop of instructions which it appears to be repeating indefinitely, or it may stop. In any case, there may be little evidence to show the original cause of the trouble. The machine operates so fast that it may have executed many hundreds of operations since the mistake was first encountered, and these operations may have obscured the evidence.

MISTAKE DIAGNOSIS

With luck, it may be possible to reconstruct the path of events merely by examining the instructions and numbers remaining in the store when the calculation ceases. In order to print these out, various small subroutines are provided with every computer, known as "post mortem" routines.

A post mortem examination can however only produce a static picture of the machine after the event. If this is insufficient, it is necessary somehow to obtain a dynamic picture of the course of action pursued by the machine during a period in which the mistake was encountered.

Most machines are equipped with manual controls which enable the operator to make the machine work slowly through a section of the programme while he examines its contents on some form of monitoring device. This however is a slow and uncertain procedure, and although various methods have been invented to make it rather more efficient, it is usually frowned upon as a procedure to be adopted by any but the most alert and experienced programmers.

Various automatic means have been devised for providing systematic information about the progress of a calculation. Some machines are equipped with special circuits which can cause them to print out automatically certain information such as a list of control transfers, transfers between fast and slow stores, etc.

Similar information can also be obtained by using an interpretive routine which regards the original programme merely as a series of parameters. It interprets these parameters according to the same code as the instruction code of the machine itself, with the exception that suitable information is printed out to assist in locating mistakes.

In some cases a combination of these techniques is used. Special circuits cause control of the machine to be switched periodically to a special checking routine which prints out or accumulates suitable diagnostic information.

PROGRAMMING NOTATIONS

In the examples given in the last lecture it was assumed that the instructions were written in the same form as that in which they were stored. It is more usual however to write instructions in a different form, and to arrange for the programme input routine to carry out a conversion when the programme is loaded into the machine. There are various reasons for this. The function part, for example, may not be easy to use and remember in its numerical form, and may therefore be written in a different code, perhaps an alphabetical one. The addresses also may need some adjustment to take care of the fact that when the programme is first written the exact positions of the various parts in the store may not be known.

This conversion of a programme during loading is only one of a series of changes which happen to it throughout its life. During the process of programming, the programme goes through a series of forms of lessening abstraction and increasing detail until it finally appears in the form of instructions. These instructions are then converted as they go into the machine, and are further interpreted in the form of a very long series of elementary operations when the programme is scanned and executed by the control unit of the computer.

Various logical changes are occurring throughout this process. Human language is being replaced by rigid symbolic language which can be recognised by the machine. General statements are being replaced by

particular statements. Short comprehensive statements are being replaced by long series of elementary ones.

Often the same logical change can be made at different stages in the development of the programme. For example, if the programmer wishes to programme a calculation on complex numbers, he may either write out the programme in terms of real numbers alone, or he may write it in terms of complex numbers and arrange for it to be interpreted suitably. In the latter case, the interpretation may be done either during the loading of the programme into the machine, each complex instruction being replaced by a short series of instructions relating to real numbers, or an interpretive routine may be used which interprets each complex instruction when it is due to be executed.

Thus, by letting his programme be translated in various ways, or by incorporating subroutines, a programmer may derive considerable benefit from the efforts of others. It is important however that all the standard routines written for a particular computer should follow similar conventions and be so designed that they can be used in conjunction with one another. A self-contained collection of notations, conventions, and available subroutines may perhaps be referred to as a "programming scheme". The more advanced types of programming scheme are sometimes referred to as "automatic programming" schemes. A well-designed automatic programming scheme can considerably reduce the amount of work involved in preparing a programme, at least for conventional types of problem. Considerable interest is currently being shown in automatic programming, particularly in the U. S. A.

The aims of a scheme are two-fold: to make programming easy to learn, and to make it easy to do when learnt. These aims are sometimes in conflict. For example, a scheme which is easy to use will employ

a fairly concise notation so that the programmer does not have an unduly large amount of writing to do. However, a concise notation is usually more difficult to learn than a more expansive one.

One of the most serious limitations in designing programming schemes is the choice of symbols which can be used. It is obviously highly desirable that the scheme shall be devised so that a programme, when written, can be typed out directly for the computer. This means that only symbols which appear on the keyboard of the equipment which is used for preparing the input to the computer may be used by the programmer.

Ideally, a programming scheme should allow a programmer to write his programme in the form in which he first states his method of solution. Recent schemes developed in the U.S.A. and in this country are certainly tending in this direction. However, as schemes become more powerful and permit the programmer to write a greater variety of words and phrases, it becomes increasingly difficult to lay down precisely what he may and may not write. It will probably never be possible to allow him to write plain English, which is usually full of ambiguities. The greater the range of things which he is allowed to write, the longer becomes the list of restrictions.

One programming task which has not yet been performed very satisfactorily by automatic means is the allocation of the space in the store of the computer or on the magnetic tape to the various items of data. Thus, for example, in a machine with two levels of storage, the arrangement of the transfers of information between the two levels can only be done satisfactorily by the programmer himself. When magnetic tape is used, the placing of the various items within one block of information on the tape must also usually be done by the programmer.