best way to start such a library is to go through existing programs and take out all the subroutines that are well written and have some general applicability (I/O routines, date routines, upper to lower case conversion, and so on). Each routine should be saved as a separate file, and these should be grouped together according to function (if they are to be stored on tape then each function group should be stored on a separate cassette) with meaningful file names to identify them. Keep a card index or a database of the file names, together with a description of what each routine does.

Needless to say, it is important to ensure that all library routines are thoroughly tested and debugged. They will be used in programs for which they were not specifically designed, so make sure that they will trap any illegal input values. You should also ensure that any values output from the library routines will not cause problems to the program that uses them. Make each routine as efficient as possible and include as much internal documentation as is necessary for you to understand the routine's function at a later date. Add to the collection as the need arises — there is no point in adding new routines 'on spec' as experience shows that this is largely wasted effort. Don't forget to number the lines of the library routines according to the convention established (this will save on RENUMbering when the routines are merged into a new program). Useful library routines may be found in computer magazines, which often publish routine listings as well as complete programs (and these can be cannibalised to obtain the useful subroutines).

To make use of a library like this, it is necessary to have a way of merging routines together to form a complete program. For those using compiled languages, a 'link-loader' or similar program is usually supplied; this takes compiled modules and joins them to make an executable program. For BASIC programmers, unless a compiler is available, the easiest way to achieve this is to use a combination of RENUMber and MERGE commands. To merge a library routine into the new program, first load the program, decide where the library routine will go and make sure there is a large enough block of unused line numbers for it to fit in. If necessary, RENUMber the library routine so that it will go into the space allotted to it. Then use the MERGE command to join the two programs; check that everything works as it should and SAVE the new program with the library routine in place.

GROUP EFFORTS

It is often the case that home computer users work together in groups to write programs — either at school or in their user clubs. Most of what has been said about program design and programmer efficiency is particularly relevant to such team efforts. In fact, most of these ideas and the concept of structured programming were developed in order to split the workload of commercial programming projects. Thus, a number of different programmers could work on different parts of the same program at the same time to produce a working program.

For BASIC programmers to work like this, it is essential to agree on the conventions to be used when coding. Assuming that a design has been agreed on, the programmer of an individual module needs to know:

1) What the files will be and how they will be organised.
2) What conventions have been agreed for naming variables. The most important variables, such as arrays that are used throughout the program, should be named in advance. A convention should be agreed for naming local variables. Variables that are passed between modules should either be named in advance or a way of ensuring that each is unique should be devised — adding the module number of the originating module as a suffix, for instance.
3) What library routines are available to the group, the format of each of these, how their variables are named, what they do, and how well tested and debugged they are.
4) How error-handling routines are organised (for instance, whether each routine copes with its own errors or whether the routines set an error 'flag', which is then dealt with by the control routine).
5) The exact function of any module that is being written.
6) The exact range and type of data that each individual module will accept as input and return as output.

This implies a lengthy planning stage with many meetings to agree strategy, followed by a short programming stage. Testing — including the testing of group-produced programs — will be dealt with later in the course. The next instalment will concentrate on the design of programs that will run faster and use less memory.

MERGE

- The Spectrum has the straightforward version of the command: it merges the named file with the program in memory; the incoming line overwrites the existing line in the event of a line number collision.
- With the BBC Micro *SPOOL command you can create ASCII versions of the program files, then write a BASIC program (or use a word processor) to access these files, one program line at a time. Merge the two files into a third ASCII file, and convert it into a program using the *EXEC command.
- On the Commodore: OPEN 1,1:CMD1:LIST:PRINT #1:CLOSE 1 creates on tape an un-named ASCII file of the program in memory. LOAD the other program, and add to it a routine to INPUT and print the ASCII file lines on screen. Stop the program and RETURN over the screen, thus merging the two programs