UNIVERSITY OF QUEENSLAND
COMPUTER CENTRE

COMPUTER CENTRE BULLETIN

Volume 3, Number 3.
2nd March, 1970.

Editor:
H. L. Smythe.

THIS ISSUE

This month's Bulletin announces the first seminar in Computer Science for 1970, gives some useful hints for debugging a program, and presents a small ALGOL puzzle. Our chief article discusses the method and some of the factors affecting the assessment of University of Queensland student fees by computer. The division in functions between the Department of Computer Science and the Computer Centre is explained, and members of the newly-created Professional Systems Development Group are introduced.

SEMINARS IN COMPUTER SCIENCE

As we mentioned in the February Bulletin (Volume 3, Number 2), the Department of Computer Science is sponsoring a series of seminars in computer science to introduce clients to new techniques in computing.

The first seminar will be presented by *Mr. Ian Oliver*, Consultant and former Departmental Lecturer, who will discuss some aspects of *Problem Solving by Simulation*. This seminar will be held on Wednesday, 18th March, in Room G13 of the Engineering Administration Building, from 2 to 4 p.m. The second seminar of the series *Paging in the Management of Core Storage* will be presented by *Mr. Michael McLean*, a Senior Demonstrator in the Department of Computer Science, on the 15th April, in Room G13 from 2-4 p.m.

All interested clients, staff and students may attend the seminars, and enquiries should be directed to the Secretary of the Department of Computer Science (Mrs. Judith Greenhill, extension 688). Any suggestions or comments should be referred to the Seminar Convener, Mr. John Williams (extension 8288).

STAFF NEWS

INTRODUCING THE PROFESSIONAL SYSTEMS DEVELOPMENT GROUP

In the last few months, the Computer Centre and the Department of Computer Science have undergone an internal reorganization. Staff responsibilities and functions have been redefined and a Professional Systems Development Group has been created so that the Centre and the Department are now functioning as separate and individual entities.

Prior to the division of responsibilities between the Centre and the Department, staff held both academic and service functions. The Professional Systems Development Group was therefore established to relieve academic staff of developmental, consultative and service activities within the Centre. The Department of Computer Science, like any other University Department, now performs purely teaching and research activities. The staff of the Computer Centre have been divided into the Professional Systems Development Group and an Operations Group, which provide, respectively, for both the development and operation of the computer installations, and day-to-day maintenance, scheduling, and data preparation duties.

Mr. John Noad, who is Senior Systems Analyst, was formerly a Lecturer in the Department of Computer Science, responsible for several subjects in the postgraduate Diploma in Information Processing. *Mr. Chris de Voil*, also a former Lecturer in the Department of Computer Science, is Systems Analyst.

Mr. Tim Kerridge, a Senior Systems Programmer, graduated with a Bachelor of Science degree from Liverpool University in 1962. In 1967, after working for the Royal Insurance Company in Liverpool, Tim came out to Australia to join the Bureau of Census and Statistics where he worked for three years. *Mr. John Row* is Systems Programmer.

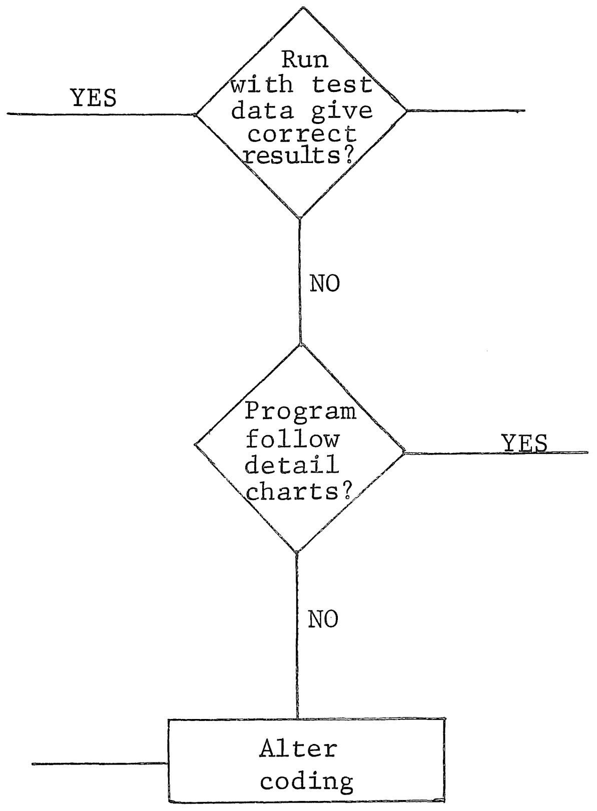
Mr. Robert Cook, who holds a Mathematics Bachelor of Arts degree and a Diploma in Computer Science from Cambridge University, and *Mr. David Seddon*, who is a Bachelor of Science graduate and holds a Diploma in Automatic Computing from the University of Queensland, have been appointed Assistant Systems Programmers.

PROGRAMMING ADVICE

A KEY TO FORTRAN DEBUGGING

J.S. Williams

Last year, the Computer Centre Bulletin (Volume 2, Number 4, pp.45-46) published a flowchart describing the systematic order in which a program should be produced. This article will concentrate on one portion of that flowchart, viz.:



YES
Run with test data give correct results?
NO
Program follow detail charts?
YES
NO
Alter coding

This *debugging* stage can often cause unnecessary difficulties. The following points are designed to help those who have mastered the fundamentals of the FORTRAN IV language, and are now writing more advanced and more complex programs.

It is important that the programmer become familiar with the writing of *subroutines* and *functions*. As a large program written in one piece is unwieldy and difficult to debug, it should be divided, in some logical manner, into several subroutines. Each subroutine should then be debugged independently of the other routines that make up the program. It is always necessary to write a test main program for each subroutine. The test main program should read in data and "set-up" values to simulate the conditions that will exist when the subroutine is called in the final working program. The subroutine should then be called, and on return to the test main program, the values calculated by the subroutine should be printed out. Great care must be taken in choosing the test data as it should test *all* possible circumstances that may arise. A program manipulating large numbers, for example, should include a test for overflow as well as a PRINT statement that prints a suitable message, should overflow occur.

Many programs involve a considerable amount of calculation before any part of the desired results is obtained. If a program is written to print only the results and no print-out occurs during a run (i.e. the program loops or stops before reaching the first PRINT statement), then the programmer has very little information to indicate the cause of the problem. It is advisable, therefore, to include several additional PRINT statements throughout the program to print out some of the important intermediate results. If the program does not run correctly, the course taken can be traced, and the values of the variables that are printed, can be noted. When the program is completely debugged, the additional PRINT statements can be removed. It is always advisable to print the data immediately after it is used (provided the volume of data is not too large). In this way, *mispunched* or *misread* cards can be detected.

A program should detect any abnormal conditions that may arise and, if no action can be taken, it should stop. Since there are usually several conditions that may arise, it is important to know the reason the program stopped. A message, therefore, should be printed, explaining the conditions existing immediately before the program stops. It is also possible that test data, apparently well-chosen, may lead to an abnormal condition. If no tests exist for abnormal conditions, the programmer will then wonder why the program appears correct but outputs incorrect results. However, if an error message is printed, the programmer can fairly readily check if it is caused by the data or yet another bug.

In a large program, it is often difficult to remember the exact purpose of each section of coding. Time is wasted if the programmer must "work" through the coding merely to discover its function. In a routine, comment cards should precede each section of coding, describing its purpose. If coding is changed at a later date, it is necessary to alter another section of coding so that the program may continue to run satisfactorily. This situation can also be noted on a comment card.

If the above points are followed, the programmer should find the debugging of large FORTRAN programs simplified. It is important that any abnormal condition that may arise as a result of the particular data used, is detected and the user is informed of the condition. It is also vital that all bugs are eliminated from the program. The user *must* be able to rely on the results he obtains.

BULLETIN BAFFLERS

Can YOU remember ALGOL?

```
begin integer a;

        procedure DOUBLA;  a:=2*a;
        a:=3;

        begin integer a;
              a:=2;
              DOUBLA;
              output(a);
              end;

output (a);

end;
```

Question: What values are output by the above code?

Next month's Bulletin will provide the answer to this *baffler*.

STUDENT FEE ASSESSMENT

A.W. Robson-Petch

The author of this article, Mr. Allen Petch, is a programmer employed by the University of Queensland, Data Processing Section. Mr. Petch has been programming for over four years, and has extensive knowledge of administrative procedures which has greatly helped in the design and implementation of programs.

This article describes the assessment of student fees for the University of Queensland. The systems analysis and programming were carried out by the Data Processing Section; and the data was processed by the GE 225 computer installed at the Computer Centre, University of Queensland.

BACKGROUND INFORMATION

The University, at present, has over 16,200 enrolments of which 7,700 are full-time, 5,200 are part-time internal, 3,100 are part-time external, and 200 are staff members. There are 200 different courses and 800 subjects are offered in twelve faculties.

The fee structure is designed so that there are two rates for full-time courses and six rates for part-time students. Certain special cases exist which do not require assessment for each term of the course. These will be dealt with later in this article.

Late enrolment fees are charged for all enrolments received after the due date, and airmail fees are imposed if external students require their class notes to be forwarded by airmail.

GENERAL DESCRIPTION

Enrolment information has been maintained on the computer for statistical and records purposes for a number of years. It was decided to extend the information contained on magnetic tape to allow for automatic assessment of fees. This tape is constantly updated with enrolment changes and hence was the automatic choice for the source information for assessment of fees.

The enrolment, fees and statistics system consists of 60 programs varying from simple cards to tape runs to the most complex update runs. The system uses exclusively the GECOM compiler and the Forward Sort Generator. It has been found that these languages are the best suited for Administrative processing under the present environment.

One of the most involved programs in the system produces each student's fee from the information contained in the updated enrolment master tape. This program runs for approximately 35 minutes and is discussed in this paper.

ASSESSMENT PROGRAM

It was found desirable to make this program as flexible as possible so that changes to course and subject codes and rates could be made without the necessity for recompilation. As compiling time for this program is about 45 minutes, it is clearly uneconomical to write fees etc. into the program.

Flexibility was achieved by using punched cards for all changeable data. A deck of 500 cards in seventeen separate layouts was necessary to contain the required input. Another method employed to reduce compiler and debugging time is to break the program into many small sections, each section dealing with one part of the entire program. If alterations of fee rules make it necessary to change the program, it is a simple task to alter the relevant section so that the remainder of the program is not affected. The data cards are stored into memory in tables as the first action performed in the program.

Normal tape label checking is done to ensure that the correct tapes are being used. If tape labels do not agree, the run is terminated. When tapes are found correct, processing commences.

Initially, enrolments are checked for cancellations. Cancelled courses are indicated on the enrolment by a special symbol, and the date of cancellation also appears. If this date is prior to the one specified in a table in memory, then a zero assessment record is produced. If not, processing continues.

One or other of two Late Enrolment Fee rates is charged if the enrolment date is later than either of the two dates stored in memory. Certain courses are exempt from this fee and are located by comparison with an exemption table.

A number of courses are assessed in certain terms of the year. If one of these is discovered by a table, look-up processing continues only if the course is to be assessed in the current term. Otherwise, a zero assessment is produced. Post-graduate courses are determined and assessed according to attendance codes.

A table, containing every course with an appropriate full-time rate code and a part-time table code, is used to identify the course for subsequent processing. When a match is made, processing continues in either of two directions.

If a full-time course is encountered, the full-time rate code is used to find the actual rate to be charged for that course. If a part-time course is encountered, the part-time table code is used to determine which table is to be used. The table of part-time set courses contains course codes and the amount to be charged for that course.

Part-time unit courses vary greatly in concept. For example, with Arts, it is necessary only to sum the number of subjects taken and multiply by the cost per subject. With Science, it is necessary to find the unit value of each subject and accumulate total units which is multiplied by the cost per unit. With Engineering, however, each subject has an associated cost, and these costs must be summed to obtain the terminal fee. Ten part-time tables exist, each requiring different means of processing. The course fees derived by one of these methods are held in a storage area for later processing.

The final stage of the run is to determine whether the fee is charged to the actual student, or to the granting body, should the student hold a scholarship. This information in coded form is contained on the enrolment tape. Separate areas are allowed on the assessment record and the fee is moved to the appropriate area prior to writing the output data.

At any stage of processing, it is possible to overrule the computer decision for assessment. An authorization to do this is shown on the enrolment record by a special indicator, and remains until deleted by a further authorization.

Automatic assessment is achieved for about 99% of all cases. When assessment is found to be impossible, whether through mispunching of relevant information or variations from the normal enrolments, a zero assessment record is produced which contains an indicator to show that this enrolment must be checked, and if necessary, a clerical assessment made.

CONCLUSION

Because of the reduction in time taken to assess students fees, assessment is now carried out four times a term, providing current figures on a regular basis.

Changes of enrolment affecting fee assessment are no longer a problem as these are dealt with automatically each time an update of the enrolment master is made.

The fees system has been operating successfully for the past three years, and only slight refinements have been necessary each academic year.