UNIVERSITY OF QUEENSLAND
COMPUTER CENTRE

COMPUTER CENTRE BULLETIN

Volume 1, No. 2.
5th August, 1968.

Editor:
J. S. Williams.

COMMENTS ON THE BULLETIN

THE FIRST EDITION

The response to the first edition of this Bulletin was very encouraging. When this edition went to press, applications for inclusion on the mailing list were still arriving. Anyone who missed the first edition and wishes to be included on the mailing list, can obtain an application form from the Secretary, Computer Centre.

THIS EDITION

This month's edition introduces some of the Computer Centre staff. It contains sections on programming advice, system modifications, and seminars, and some contributions by various members of staff. In particular the program library is described, a useful programming technique is presented, and a discussion of the role of the Computer Centre in computer education appears.

STAFF OF THE COMPUTER CENTRE

In order to acquaint clients with the staff of the Computer Centre, the Bulletin will contain a series of articles detailing their position, duties, and responsibilities. The first in this series deals with the Administrative Officer and the Computer Operators.

ADMINISTRATIVE OFFICER

The Administrative Officer, Mr. John Jauncey, is responsible for computer services, and all functions related thereto. His office is situated in the lobby of the Computer Centre. Mr. Jauncey receives requisitions from University departments and maintains detailed accounts of expenditure by all clients. Blocks of computer time longer than 15 minutes may be booked with him. As well as supervising all computer input and output, Mr. Jauncey is responsible for data preparation and the program library.

Any enquiries relating to Computer Centre operations or services should be directed to the Administrative Officer, (extension 8471) in the first instance.

COMPUTER OPERATORS

The Computer Centre is open to clients from 7 a.m. to 12 midnight, but the equipment is actually in operation from 9 a.m. till 7 a.m. the following morning. Preventive maintenance is carried out between 7 a.m. and 9 a.m.

Six female computer operators are currently employed on the day and afternoon shifts to operate the GE 225 (and eventually the PDP 10). They are Ann McArthur, Pat Loder, Pat Matthews, Helen Otte, Noela Sparke, and Diann Wilkins. A seventh operator, Mr. Pat Cusack, who is a part-time student, works permanent night shift from 11.30 p.m. to 7.30 a.m.

The operators are responsible for operating the computer in accordance with the instructions contained on the user's Run Request Card, or the appropriate library writeup. Efficient operation requires manual dexterity, attention to detail, and a considerable amount of concentration on the task in hand. For this reason, distractions in the nature of changes to card decks or answering client's queries are not encouraged. Therefore, all queries relating to operation of the computer must be addressed to the Administrative Officer, extension 8471.

SEMINARS AND PROGRAMMING COURSES

SEMINAR: *A Compiler for Continuous Systems Simulation*

The speaker will be Mr. L. Mor, a postgraduate student in the Department of Electrical Engineering. The seminar will be between 12 and 1 p.m. on Wednesday, 28th August, 1968, in Room B18 of the Engineering School.

The solution of time-varying linear differential equations is often carried out on an analogue computer. As part of a M.Eng.Sc. Thesis, the speaker is writing a problem-oriented language for the simulation of analogue computer techniques using a digital computer. The language combines the advantages of both analogue and digital computation techniques.

For further particulars concerning this, or future seminars, please contact the Secretary, Computer Centre, extension 688.

FORTRAN IV COURSE

A FORTRAN IV course is to be held from Monday 19th to Friday 23rd

August, inclusive, between the hours of 9 a.m. and 1 p.m., in Room 214 of the Engineering School, Circular Dirve, St. Lucia.

A large number of applications had been received when applications closed. Unfortunately enrolment is limited to 25 for the course and some applications will not be accepted. Details of subsequent courses will be promulgated at a later date.

PROGRAMMING ADVICE

FORTRAN PROGRAMMING ERROR UNMASKED!

An error fairly frequently made in CARD FORTRAN and FORTRAN IV programming is illustrated by the statement

```
      CALL SQUARE (PEGS, 5)
```

where PEGS is an array of length 20 and the subroutine SQUARE is as follows:

```
      SUBRØUTINE SQUARE (RØUND, N)
      DIMENSIØN RØUND(20)
10    RØUND(N)  =  RØUND(N) **2
      N  =  N + 1
      IF (N - 20) 10, 10
      RETURN
      END
```

In the subroutine, N is only a dummy argument. This means that the actual argument in the main program, in this case the constant 5, is referenced by the subroutine at every appearance of N. The replacement statement N = N + 1 therefore causes the value of the constant 5 to be incremented by one. On exit from the subroutine the main program will use the value 21, the final value generated by the subroutine, wherever the constant 5 appears.

In general, the rule is that if a dummy argument in a subprogram is changed within the subprogram, then the corresponding argument in any reference to the subprogram must not be a literal constant (such as 1, 2.5, etc.)

To avoid this problem the subroutine may be modified as follows:

```
              SUBRØUTINE SQUARE (RØUND, N)
              DIMENSIØN RØUND(20)
              K  =  N
      10      RØUND(K)  =  RØUND(K)**2
              K  =  K + 1
              IF (K - 20) 10, 10
              RETURN
              END
```

DOUBLE PRECISION AND COMPLEX LIBRARY FUNCTIONS IN FORTRAN IV

It is not universally known that in GE 225 FORTRAN IV the name of *library* (as well as user-defined) functions which return complex or double-precision values have to be mentioned in appropriate type declaration statements, thus -

```
              SUBRØUTINE FIGURE (A, B, C, D)
              DØUBLE PRECISIØN A, B, DABS, DSIN
              CØMPLEX D, CCØS
              A  =  DABS (DSIN (B))
              C  =  CABS (CCØS (D))
              RETURN
              END
```

In this example, the library functions DABS (double-precision absolute value) and DSIN (double-precision sine) both return double precision values, and so must be named in the DØUBLE PRECISIØN declaration. Similarly, the library function CCØS (complex cosine) is named in the CØMPLEX declaration because it returns a complex value. However, the library function CABS returns a real-valued result, not a complex one. (The absolute value of a complex number is real.)

MCML II LOADER

Care must be taken in the use of octal corrections with the MCML II loader. Octal corrections may not be stored in locations having addresses greater than the highest address of the routine, because octal corrections do not update the loader's current address counter. A patch area must be provided for this purpose.

```
                ...
                ...
                ...
        PATCH   BSS      19
                DEC       0
                END       0
```

Note that in a relocatable routine which is loaded by MCML II, a BSS instruction should never be the last instruction requiring storage to be allocated. Using the combination BSS, DEC in the above manner sets aside 20 locations for use with octal corrections.

SYSTEM MODIFICATIONS

CHANGES TO CARD FORTRAN EXECUTION ERROR MESSAGES

If the argument X of the library function ACØSF (inverse cosine) is not in the range $-1 \leqslant X \leqslant 1$, the result is set to the largest number (10^{76}), the error message "ACS" is typed, and the execution continues. The procedure is the same for the inverse sine function ASINF, the error message in this case being "ASN".

NEW FORTRAN IV EXECUTION ERROR MESSAGE

An additional message has been added to the FORTRAN IV execution routines to detect an attempt to read an end-of-file card.

"EFC" Attempted READ encounters an end-of-file card. Execution stops.

NEW FORTRAN IV TEST FUNCTION

The function IEØFX tests the card next to be read for the end-of-file card image. If it is, the function returns the value 'true' and the card is skipped. If not, the function returns the value 'false'.

The standard FORTRAN IV function calling sequence is used -

N = IEØFX(M)

where M is a dummy parameter, used only to identify the function to the compiler as an external function. Any variable name or constant may be used, since it is not changed by execution of the function.

This function is similar to the IEØF function included as part of the tape routines but, since it only tests the card reader for the end-of-file condition, it occupies considerably less storage space.

LIBRARY PROGRAMS

NEW

HDIAG - Eigenvalues, Eigenvectors (D4.221)

This FORTRAN IV subroutine has been converted from CARD FORTRAN, and is used in the same manner as in CARD FORTRAN. It calculates the eigenvalues, or eigenvectors and eigenvalues, of a real symmetric matrix.

REPLACEMENT

QSIMEQ - Linear Simultaneous Equations (D4.202)

This is a complete program which will obtain the solution of a set of up to 54 simultaneous linear equations. SLIP is used for input, thus quotes must enclose the identification information on the IDENT card.

RECENT PUBLICATIONS

The following publication is available at the Computer Centre:

Technical Memorandum No. 3: *Macro Facilities in GAP.* W.N. Fulton.

The memorandum describes the new macro facilities implemented by Mr. Raife Eldershaw, that are now available in the GE 225 assembly language, GAP. Macro instructions can be quite useful when blocks of coding of similar structure are needed at several points in a single program.

PROGRAM LIBRARY - J. Jauncey

The program library at the Computer Centre is maintained for the use of clients as well as to ensure the smooth functioning of the Centre itself. Contained in the library are master copies of all programs, manuals, listings, etc., and full details of the holdings are listed in the Program Library Status Report, the latest edition of which is 1st May, 1968. These reports are available upon request and copies of most of the material may be obtained. A copy of all material is available in the clients room for reference purposes. In some instances, writeups on frequently used programs are readily available.

CLASSIFICATION

Each program incorporated in the program library is classified according to the program function and then with respect to its origin. The program function categories and sub-categories are as follows:

A - Administrative Programs

B - Utility Programs

- 1 Memory Loaders
- 2 Memory Dumps
- 3 Tape Dumps
- 5 Traces and Debugging Routines
- 6 Miscellaneous Card and Tape Programs
- 7 Symbol Converters and Relativisers
- 8 Analysers

C - Internal Data Manipulation

- 1 BCD to Binary
- 2 Binary to BCD
- 3 Octal Routines
- 4 Number Conversion
- 5 Internal Sort, Search
- 6 Internal Data Movement
- 7 Character Conversion
- 8 Miscellaneous

D - Mathematical Routines

- 1 Programmed Arithmetic
- 2 Elementary Functions
- 3 Statistical Analysis and Probability
- 4 Operations on Matrices, Vectors, and Simultaneous Linear Equations
- 5 Polynomial and Special Functions
- 6 Curve Fittings, Interpolation, and other approximations
- 7 Operations Research
- 8 Numerical Integration, Differentiation, and Solutions of Differential Equations

E - Input/Output

- 1 Card
- 2 Magnetic Tape
- 4 Paper Tape
- 5 Console Typewriter
- 6 High Speed Printer
- 8 Mass Random Access File
- 9 Special Devices

F - Assembly Systems

G - Generators

H - Compilers/Translators

1 GECOM
2 WIZ
4 Special Purpose Languages (including MAC, WIZOR, WISP)
5 COBOL
6 Algorithmic Languages (including CARD FORTRAN, FORTRAN IV)

I - Simulators

J - Service Systems

K - Special Systems

M - Electric Utility Programs

O - Machinery Control Systems

P - Railroad and Road Systems

Q - Accounting Programs

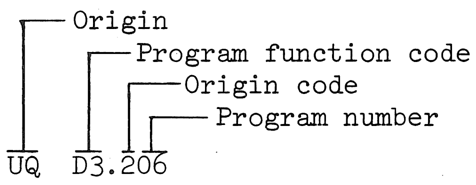
Programs contained in the library originate from the following three sources:

CD - Computer Department, General Electric (origin code 0)

GET - General Electric 225 Users Association (origin code 1)

UQ - University of Queensland (origin code 2)

A library number is therefore constructed in the following format:



Since many programs perform similar functions, they have similar names, therefore references to programs in the library (with the exception of loaders) should be by program number.

USE OF LIBRARY PROGRAMS

It is unnecessary for a user to obtain a copy of a program card deck in order to use the program. He need only submit his data, set up in accordance with the specifications contained in the writeup, and insert the program number on a Run Request Card in the section labelled "Std. Programs Required, Other". The remainder of the run request card should be completed in the normal manner.

However, if a copy of a program deck is required, a Run Request Card should be submitted, with the section REPRODUCE/COPY marked, as well as the library number of the program deck required.

Any queries concerning the program library should be directed to the Administrative Officer.

COMPUTER EDUCATION

A computer system which is functioning normally is not newsworthy, so the occasional error which is made often results in adverse publicity. Understandably, the public image of computers leaves much to be desired. This adverse publicity is the consequence of the mistakes made by people rather than by computers.

Unfortunately the computer field is expanding so rapidly that the number of suitably qualified people is completely inadequate and many of the mistakes are caused by inadequately trained personnel.

Training may consist of formal courses, in-service training, or the obtaining of ad hoc experience. Unfortunately, the training of many computer people has been composed of the latter. The State and Commonwealth Governments and a few of the computer manufacturers have actively promoted in-service training, while the universities, and more recently the colleges of advanced education, have established formal courses. As with most other professions, training, or the learning process, must extend beyond the formal course which merely enables a person to enter his chosen field.

The computer field is advancing and expanding with such speed, that there is an urgent need for the professional computer people to be exposed to all three forms of education. A basic understanding of the fundamentals must be acquired first, so that the student has a base on which to build. This must then be supplemented by continuing in-service training as new techniques are developed, the acquiring of on-the-job experience, and reading of the professional journals.

The following sections outline some of the ways in which the University of Queensland is contributing to computer education.

SOWING THE SEED - R.N. Buchanan

The continued development of information processing and computer science is dependent upon an increasing number of graduates in these fields and, by implication, a sufficient number of secondary school students who are interested in careers in these fields. The reader is probably closely involved with computers and is very familiar with the challenge and excitement of this type of work. However, it is quite likely that there are hundreds of students in high schools who are potential programmers, systems analysts, or computer scientists, but who have never given a thought to these professions merely because they have no idea of the type of work involved and are unaware of the avenues by which they may enter the field.

It appears that an effective method of interesting secondary school students is per medium of the University's program of careers lectures in high schools. A member of the Computer Centre staff has been visiting the various metropolitan high schools and discussing the different courses available and the associated career opportunities. To date, lectures have been given at eight schools involving over 2,500 students. The interest generated indicates that there should be a future supply of students for courses offered by the University.

POSTGRADUATE DIPLOMA IN INFORMATION PROCESSING - J.D. Noad

Over the past few years several surveys of electronic data processing installations in Australia have been made in an attempt to estimate the number of qualified systems analysts and programmers required in future years. While estimates from different sources vary considerably, all estimates do show that there is a requirement for more trained people than are presently available, and that with the present rate of training the situation is likely to get worse rather than better in future years.

In an attempt to help meet this need the University this year introduced a postgraduate Diploma in Information Processing. This one-year full-time course is run within the Faculty of Commerce and Economics in conjunction with the Computer Centre.

The course is oriented towards the design and application of computer-based information processing systems and gives particular emphasis to electronic data processing systems and decision-making in the business and governmental fields. It covers the functions and use of data processing equipment, computer programming and programming techniques, some basic mathematics, the analysis and design of information processing systems, and the application of computers to economic information processing and managerial decision making.

In addition to the basic theory of systems and programming, particular emphasis is placed upon the application of computers to problems of business and industry, involving the student in a number of practical exercises. During the course, computer programs, using several different programming languages, have to be written, and other program packages are utilized. The practical work in systems analysis and design is carried out by the student in a realistic environment, including a number of semi-governmental and commercial organisations.

With the rapid expansion of EDP into nearly all major fields of commerce, industry, and government, it is felt that this course is fulfilling a need for all graduates planning careers in these areas.

SEMINARS IN COMPUTER SCIENCE - I. Oliver

The Computer Centre is sponsoring a series of seminars in the Computer Sciences which may be of interest to a broad cross-section of the computing community. The purpose of these seminars will be to make known some of the newer techniques in computer science both from a purely computing and from an applications viewpoint.

Although the seminars will be fairly technical in nature and will pre-suppose some general knowledge of computing, each speaker will presume that the audience has no prior knowledge in the specific subject area.

Contributions to this series are being solicited. Any suggestions concerning possible topics or speakers would be much appreciated and should be directed to Mr. Ian Oliver, extension 575.

First Seminar: *WISP - A New Programming Language*

This seminar was held on 31st July, 1968, and was well received by an audience of more than 50 people. Mr. Bill Fulton or Mr. John Williams will conduct a WISP programming course early next year if there is a sufficient demand. Applications should be made by telephoning the Secretary, Computer Centre, extension 688.

USEFUL PROGRAMMING TECHNIQUES

VARIABLE DEPTH NESTING OF DO LOOPS - I. Oliver

Have you ever had the problem of trying to program a nest of DΘ statements in which the depth of the nest is variable? The sort of thing you might want to do is

```
      DΘ   10  I1  =  1, N1
      DΘ   10  I2  =  1, N2
      ...
      ...
      ...
      DΘ   10  Im  =  1, Nm
```

Of course FORTRAN won't accept a series of dots as an indication that m DΘ statements are required. A different way of programming this according to the rules of FORTRAN is needed.

What we would like to do is to put the DΘ statement itself in a DΘ loop, thus -

```
          DΘ   20  J  =  1, M
20        DΘ   10  I(J)  =  1, N(J)
```

but this is illegal, since the range of the DΘ 20 overlaps the range of the DΘ 10. In addition, we cannot use subscripted variables as indices in a DΘ statement.

The following statements will do the job:

```
          DΘ   10  J  =  1, M                 (initialization)
10        I(J)  =  1
20        ...                                 (calculation)
          ...
          ...
          DΘ  30  J  =  1, M
          I(J)  =  I(J) + 1                   (increment)
          IF(I(J).LE.N(J)) GΘ TΘ 20           (test)
          I(J)  =  1
30        CΘNTINUE
```

N is an array of elements containing the maximum values of the indices. Note that this complex loop contains the same basic elements as a simple loop - initialization, calculation, incrementing and testing. The DΘ 10 sets the elements of an array of indices, I, to their initial value 1. The statements which would appear at statement number 20 perform the desired calculations, presumably making use of the index array. The DΘ 30 loop increments index I(1) first every time. Provided this is not greater than N(1) control passes to statement 20. When I(1) is greater than N(1) it is reset to 1 and the second index I(2) is incremented by 1. Not until I(1) has been incremented up to N(1) again does I(2) get incremented again. Finally I(2) will reach N(2) and then I(3) will be incremented. In this way all possible combinations of values of the indices in the array I will be generated in turn.

This complex loop is actually similar to the DΘ statements in the first example in the reverse order. Index I(1) is varied most rapidly and I(M) least rapidly.

This technique was used in a problem in which an array with a variable number of dimensions was required - another thing FORTRAN does not allow. For example, in one run a three-dimensional array was required, and elements like X(I, J, K) were needed. In another run the array had five dimensions and access was required to elements such as X(I1, I2, I3, I4, I5).

The solution to this problem was to declare the array X as a one-dimensional array and to refer to locations such as X(J), where J is calculated by a function subprogram in the following manner: J = INDEX (I, N, M). In other words, the function INDEX is written so as to compute a single subscript value from the array of M indices I. The array N contains the corresponding upper limits of each index.

Let us suppose we want to put zero in every location of the M-dimensional array X. Then we use the complex loop described above with the following statements inserted at statement 20 -

```
20      IND  =  INDEX(I, N, M)
        X(IND)  =  0.
```

The function subprogram INDEX is written as follows -

```
        FUNCTIΘN INDEX (I, N, M)
        DIMENSIΘN I(M), N(M)
        INDEX  =  0
        DΘ 10 J  =  1, M
```

```
10      INDEX  =  INDEX*N(J) + I(J) - 1
        INDEX  =  INDEX + 1
        RETURN
        END
```