

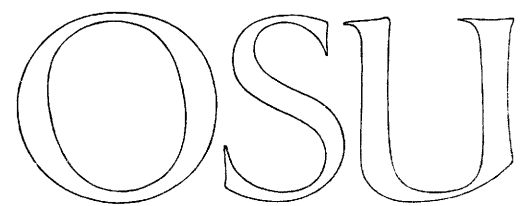
# **New Version of OSCAR**

(Addendum to "A Brief Description  
of OSCAR," cc-68-45)

by

**Gilbert A. Bachelor**

**March, 1969**

The logo for Oregon State University, consisting of the letters 'OSU' in a large, outlined, serif font.

**COMPUTER CENTER**

Oregon State University  
Corvallis, Oregon 97331

NEW VERSION OF OSCAR

(Addendum to "A Brief Description of OSCAR" cc-68-45)

by

Gilbert A. Bachelor

cc-69-6

Computer Center  
Oregon State University  
Corvallis, Oregon 97331

March, 1969

## New Version of OSCAR

A new version of OSCAR (V54) has been released. There are a number of changes and improvements over version 53. These are described below.

### Changes

1. Formerly, the character @ could be used in data input (READ statement) to indicate that the variable being read should not be changed. This has been changed; the dollar sign (\$) is now used instead of @.
2. The READCHAR statement will now give an empty character string as a result when either a quote mark (") or a carriage return (end of line) is read. Formerly, only the quote mark produced this result. In the next version of OSCAR, READCHAR will give an empty character string only on end-of-line, while the quote mark will produce a character string containing a quote mark.
3. The handling of long symbols in & commands has been changed. Previously, if a symbol had more than 8 characters, OSCAR would use the first 7 characters and the last character. This caused problems; in commands such as &SAVE,20=ZORCHDATA, OSCAR would save unit 20 under the name ZORCHDAA. However, all other systems under OS-3 that handle file names use the first 8 characters of long names. So, an OS-3 statement #EQUIP,10=ZORCHDATA would try to equip a file name ZORCHDAT, which is obviously not the same as ZORCHDAA. To eliminate this incompatibility, OSCAR V54 now uses the first 8 characters of long names in & commands.

Improvements and Correction of Bugs

1. ENTIER and FP used to give incorrect results for certain values. These bugs have been fixed.
2. The dollar sign (\$) is now allowed as a substitute for semicolon (;). Either of these characters may be used to separate statements. This change was made for the convenience of users of display consoles, since the consoles do not have the semicolon.
3. When an error occurs in an & command in a stored program, the error messages will now indicate which step was being executed.
4. The OSCAR translator can now handle longer lines. Previously, if a line produced a translated form of more than 63 items, the message TOO LONG OR TOO COMPLICATED would be printed. The translator can now handle a line of up to 127 items.
5. There was a problem with having too many references to constants. For example, a statement such as A(1:25, 1:25):=1 would produce the message REFERENCE COUNT OVERFLOW, because this statement would result in 625 references to the constant 1. The reference count was (and still is) limited to 511. In OSCAR V54, short constants (4 characters or less) effectively no longer have reference counts, so any number of references to such constants are allowed.
6. When OSCAR is reading information from files, records must not be longer than 136 characters. Previously, if a longer record were read, OSCAR simply truncated it to 136 characters. OSCAR V54 complains about long records; it prints the message INPUT RECORD TOO LONG.

### Extensions to Existing Features

1. Previously, there were limitations on the use of zero as a subscript.  $A(0)$  was allowed, but  $A(2,0)$  was not. Also,  $A(0)$  was treated essentially as a separate variable and was not included in the array  $A$ . This has been changed so that zero subscripts are now allowed in any subscript position, and the elements with zero subscripts are included in the arrays. Operations on arrays will now include elements of zero subscript. However, one cannot invert an array with such elements ( $1/A$  will fail). Subscript range is now limited to either 0 to 254, or to 1 to 255 (user's choice).

When arrays are PRINTed, no indication is given as to whether the subscripts start with 0 or 1. However, if an array is EPRINTed, the word ARRAY means that subscripts start with 1; the word ZARRAY means that they start with 0. One can also use the word ZARRAY when typing in an array, to specify that subscripts should start with 0.

2. Sub-array notation has been generalized, to allow any number of levels. For example,  $B(4:7,1:5,4,0:6)$  denotes the sub-array of  $B$  that contains the elements in which the first subscript ranges from 4 to 7, the second subscript from 1 to 5, the third subscript is 4, and the fourth subscript ranges from 0 to 6. Also, a notation such as  $A(3:2)$  is now allowed. This produces an "empty" result that does not affect values of other quantities combined with it. This feature should simplify the programming of certain types of calculations. For example,  $A(I+1:N)$  might occur in a loop where  $I$  ranges from 1 to  $N$ . Previously, one would have to program a check to do something special when  $I=N$ , to prevent an error from occurring on  $A(N+1:N)$ . In the new scheme, such testing should usually be unnecessary.

3. Previously, the ARG function was defined only for two real arguments: ARG(X,Y) is the angle ( $-\pi$  to  $+\pi$ ) between the positive x-axis and the line from the origin to the point (X,Y) in a cartesian coordinate system. In OSCAR V54, ARG is also allowed as a function of a single complex number; ARG(Z) is the angle between the positive x-axis and the vector representing the complex number Z. That is,  $\text{ARG}(Z) = \text{ARG}(\text{RE}(Z), \text{IM}(Z))$ .

### New Features

1. The determinant function DET has been implemented. DET(A) has a value which is the determinant of the square array A.
2. The functions EPART (exponent part) and NPART (number part) have been implemented. For example, if  $X = 1.2345E21$ , then  $\text{EPART}(X) = 21$  and  $\text{NPART}(X) = 1.2345$ .
3. Character string processing in OSCAR is being changed considerably. These changes are not present in OSCAR V54, but the words DECODE, ENCODE, and TEXT are now reserved words (not yet implemented), in preparation for this change.
4. OSCAR can now be used at display consoles as well as from teletypes. The CRT user types  $\neq$ OSCAR and presses SEND. OSCAR will display some instructions, indicating differences between teletype and display usage. The input/output routines for display usage occupy some storage space, so the CRT user has less storage space than a teletype user. The experimental version \*TVOSCAR is being withdrawn.
5. A new command has been implemented:  $\&\text{WIDTH}, \langle \text{integer} \rangle$  specifies the number of characters per line on the user's console. The WIDTH is preset by OSCAR to 72 for a teletype, or 50 for a display console. The  $\&\text{WIDTH}$  command

allows the user to change the standard width, if he so wishes. This is especially useful when the user's console has a longer or shorter line than usual. Note: the WIDTH command has no effect on the length of records written on files or other output devices. These record lengths are determined by OSCAR.

6. OSCAR V54 has a new feature that allows the user to define new operators of his own choosing. To do this, the user first picks a name for the operator. The name must start with a question mark (?), and may include up to 3 letters and/or digits after the (?). For example, one could use the following as names for operators:

?    ?A    ?3X    ?ADD    ?42

An operator is defined by a LET statement, and must have two arguments. For example:

LET ?Q(X,Y) = SQRT(X<sup>2</sup> + Y<sup>2</sup>)

An operator so defined is then used between its operands, like other operators. For example, B:=3 ?Q 4 will assign the value 5 to B. The statement C:=3 ?Q 4 ?Q 12 will assign the value 13 to C. (3 ?Q 4 is 5, and 5 ?Q 12 yields 13).

All ?operators are considered to have the same precedence, which is higher than all other operators except %.