Department of Applied Analysis
and Computer Science
Technical Report CSTR 1011
August 1971

WATFIV STRING MANIPULATION ROUTINES

by
Doron J. Cohen
L. Bryan Douglas & Sheila F. Fearn

WATFIV STRING MANIPULATION ROUTINES

CONCAT,SUBSTR and BOOL are WATFIV subprograms and INDEX is an integer function which have been written in Assembler/360 language to be called from FORTRAN programmes. These subprograms perform the operations identical to the PL/1 functions of the same names but the WATFIV subprograms and INDEX are generic, meaning that the arguments are not restricted to one particular data type but a combination of any valid WATFIV data types can be used. For example, an array and a character string can be concatenated by the CONCAT routine.

CONCAT will concatenate any valid data types and place the result in a specified field. Because the data types can be combined in any manner, CONCAT can be used to perform operations which are otherwise unavailable in WATFIV. The routine does not check for undefined variables but it is the user's responsibility to ensure that he uses it correctly.

The function INDEX searches through a specified variable or array to find a match with the string specified. The return value from the function is INTEGER*4 and is equal to the subscript of the first element of the array where a match was found; or if the source is character then is

equals the number of bytes into the source where a match was made. The value 0 is returned if a match is not found.

The program SUBSTR is used to extract a specified substring from a given string or array and places it in a substring of another string or array; hence SUBSTR can perform assignment into a substring of a variable.

BOOL performs one of sixteen logical boolean operations indicated by the function parameter on the specified arguments. The arguments, as in the other functions can be any valid WATFIV data types but the one restriction is that the total length in bytes of the arguments must be equal.

The four routines are described in greater detail on the following pages. The calling sequence is specified and tested examples are included to aid the programmer in using these routines.

CONCAT

This is a WATFIV routine to perform the concatenate function. The elements to be concatenated can be be any WATFIV data types; and the result field can be any data type which can be legally changed.(ie. excluded are constants, do-parameters, etc.) Data types can be mixed in any manner imaginable so the routine can be used to do moves which are otherwise impossible in WATFIV.

The calling sequence is as follows;

```
CALL CONCAT(SOURCE1,SOURCE2,RESULT [,LENGTH1 [,LENGTH2]] [,ERRET])
```

SOURCE1,SOURCE2 and RESULT may be either simple variables, arrays or array elements. SOURCE1 and SOURCE2 may also be constants. LENGTH1 and LENGTH2 are optional but if specified must be INTEGER*2 or INTEGER*4. If the lengths are not explicitly specified they will be assigned the default lengths as described below. Obviously it is not possible to omit LENGTH1 and explicitly declare LENGTH2 so if either length is negative it will be ignored and the default length will be used. If either of the explicit lengths is zero the corresponding source will be ignored.

ie. `CALL CONCAT(5,1,I,1,0)`

would be equivalent to I=5. EPRET is also optional but it may be specified even if one or more of LENGTH1 or LENGTH2 are omitted. (EPRET is written as &1 where i is the statement number to which control will be returned.)

The function of the routine is to place SOURCE2 immediately after SOURCE1 and place this result in RESULT.

```
EXAMPLE 1.    CHARACTER FN*4/'JOHN'/,NAME*8/'DOE      '/
              CHARACTER NAMES*8
              CALL CONCAT(FN,NAME,NAMES)
              PPINT,NAMES
```

In this example, we concatenated the two character strings FN and NAME and put them in the result field NAMES. The field now contains 'JOHN DOE'.

```
EXAMPLE 2.    INTEGER*4 A(2)
              CALL CONCAT(5,6,A)
   is equivalent to
         A(1)=5
         A(2)=6
-------
EXAMPLE 3.     REAL*4 A(5),B(5)
               CALL CONCAT(B(2),B(5),A(2),2,1)
   is equivalent to
         A(2)=B(2)
         A(3)=B(3)
         A(4)=B(5)
```

Lengths specified always refer to array elements except in the case of character variables where lengths are always in bytes.

In EXAMPLE 2 we concatenate the integer 5 to the integer 6 and place the result in the vector A. Since no lengths were specified the defaults are 1(since both 5 and 6 are simple constants) and so the result is 2 words.

In EXAMPLE 3 we take 2 elements starting at B(2) and concatenate them to 1 element at B(5). The result is therefore B(2) B(3) B(5) and this is placed in the RESULT field which starts at A(2). Note that A(1) and A(5) will remain unchanged.

The default lengths are calculated as follows;

For a simple variable the length is 1.

For an array the length is the number of elements in the array.

For an array element the length is the number of elements remaining before the end of the array.

ie. for INTEGER A(10)

the default length of A(8) would be 3.

For character variables the length is the total number of bytes in the variable or array.

If LENGTH1 and/or LENGTH2 are specified they are checked against the default length and if larger, execution is terminated with a FN-4 error. If the sum of the lengths(in bytes) of SOURCE1 and SOURCE2 is greater than the length(in bytes) of RESULT then it will be truncated to fit in RESULT(unless ERRETURN is specified in which case control returns to it, before assignemnt is made to the RESULT field).

Because CONCAT does not check that the data types match it can be used to move data into a variable of a different data type.

EXAMPLE 4. CALL CONCAT(0,1,A,0)
is equivalent to

```
REAL*4 A
INTEGER*4 B
EQUIVALENCE(A,B)
B=1
```

Although A is REAL it now contains the integer representation of 1. The subroutine may also be used to zero an array in one call in the following manner.

CALL CONCAT(0,A,A,1,N)

where N is one less than the number of elements in A or one less than the number of elements which it is wished to zero. 0 is moved to A(1),A(1)±now 0] is moved to A(2),A(2) now 0] is moved to A(3) etc. In this way an entire array or any part thereof may be set to any given value.

EXAMPLE 5.

```
CHARACTER*2 A(5)/'XX','HE','LL','O ','XX'/
CHARACTER*10 B
CALL CONCAT(A(2),'DERE',B,6)
```

B is now equal to 'HELLO DERE'.

EXAMPLE 6.

```
CHARACTER*14 A/'  JOHN  DAVID '/
CHARACTER*6 B/'HENRY '/
CHARACTER*13 C
CALL CONCAT(A,B,C)
PRINT,C
```

In this example the SOURCE1 and SOURCE2 are longer than the result field, so they are truncated to fit in the field C. C now contains 'JOHN DAVID'.

```
EXAMPLE 7.          CHARACTER*14 A/'  JOHN  DAVID '/
                    CHARACTER*6 B/'HENRY '/
                    CHARACTER*18 C
                    CALL CONCAT(A,B,C,&1  )
                    PRINT,C
                1   PRINT2
                2   FORMAT('RESULT FIELD IS TOO SHORT')
```

In this case, the result field is again too short for SOURCE1 and SOURCE2 but ERRET has been specified so control is passed to this statement and the message is printed out.

NOTE:

This subroutine does a move directly from the source to the result(left to right, byte by byte). Although this has its uses in propogating something through an array it can be dangerous if not taken into account. Thus if the source and result are elements of the same array and there is overlap between them, the result may not always be that which was expected. Remember that if something is placed in an element of the result which overlaps the source and then that element is used as source it will be the new value which is placed in the result, not the old value.

ALSO NOTE:

If elements of the source are undefined the corresponding elements of the result will then be undefined, even if they had been defined previously.

INDEX

This is a WATFIV function which will scan through a source field and attempt to find a match with a given pattern. The calling sequence is as follows;

I=INDEX(SOURCE,PATTERN [,LENGTH])

The source may be a character string or any type of array, the only restriction on the pattern is that the length of each element must be the same as the length of each element in the source. ie. if the source is REAL*4 the pattern may be REAL*4, LOGICAL*4 or INTEGER*4 (not CHARACTER*4 however, since character variables are treated on a byte basis). If the source is character then the pattern must be character but it makes no difference if the strings are of different lengths or are arrays, as long as they are character. The return argument from the function is INTEGER*4 and is the subscript of the first element of the array which was equal to the pattern. If the arguments are character variables then the function value is the number of bytes into the source If the pattern is not found in the source then the function value will be zero.

EXAMPLE 1.

```
CHARACTER*2 A(3)/'AB','CD','EF'/
I=INDEX(A,'D')
```

I=4 since 'D' was the 4th byte of A.

If the pattern is an array element only the one element will be used for the pattern unless the optional length is specified. If an array name is used for the pattern then the entire array will be used. If the pattern is a character string then a match will be sought on the entire string, if it is an element of a character array then the rest of the array from the element specified until the end will be used. The length if specified will be in array elements (bytes if a character string). If the length specified is longer that that available in the array or string then a warning message will be issued and execution will continue using the given length.

NOTE:

The explicit length if used must be either INTEGER*2 or INTEGER*4, anything else will be ignored and the default length used.

The following examples should help illustrate the use of this function.

EXAMPLE 2.

```
INTEGER*2 A(10)/1,2,3,4,5,6,7,8,9,0/
I=INDEX(A,8)
```

This program will terminate with a FN-5 INVALID PARAMETER since 8 will generate a INTEGER*4 variable and A is INTEGER*2.

EXAMPLE 3.

```
INTEGER*4 A(10)/1,2,3,1,2,3,4,8,9,0/
INTEGER*4 B(4)/1,2,3,4/
I=INDEX(A,B(1))
J=INDEX(A,B)
```

I=1 since it will only look for the first element of the pattern.

J=4 since it looks for a match on the entire pattern and does not find this until the 4th element of the source.

EXAMPLE 4.

```
CHARACTER*10 A/'1231234567'/
I=INDEX(A,'1234')
J=INDEX(A,'1234',3)
```

I=4 since it uses the entire character string and does not find a substring of A that matches until the one which starts with the 4th byte.

J=1 since we use only 3 bytes and a match is found at once.

EXAMPLE 5.

```
CHARACTER A*4/'1234'/,B*2(3)/'25','34','71'/
I=INDEX(A,B(2))
```

I=0 since B is a character variable it uses the rest of the array for the pattern and a match is sought on '3471', if you wish to search only for the 2nd element you should code

```
I=INDEX(A,B(2),2)
```

this will look only for the 2 bytes which start at B(2).

SUBSTR

This program combines the functions of the PL/1 SUBSTR function and pseudo-variable. That is to say that it can be used to extract a specified substring from a given string or array and to place it in a specified substring of another string or array. The calling sequence for this routine is as follows:

CALL SUBSTR(SOURCE,RESULT,SDISP,LENGTH [,RDISP] [,RETURN])

SOURCE is the field we are going to move from, RESULT is the field we are going to move to. SDISP is the number of the element in the source where we want to start the move (element will be explained later). LENGTH is the number of elements that we want to move. RDISP is optional and if specified is the number of the element in RESULT where we start placing the source field. RETURN is optional and if specified is the statement number to which we will return if the length and displacement specified do not fit in the SOURCE. (the RETURN parameter would be written as &s where s is the statement number to which control will be returned).

SOURCE and RESULT can be character strings or any valid WATFIV arrays but they must have elements of the same length . An element is an array element if SOURCE and RESULT are arrays of any other type than CHARACTER. If SOURCE and RESULT are character variables of any type (strings or

arrays) then an element is considered to be 1 character. Thus if SOURCE was REAL*4, then RESULT could be INTEGER*4 but not CHARACTER*4 (it could in fact only be INTEGER*4, REAL*4 or LOGICAL*4 since there are no other data types with elements which are 4 bytes long).

```
EXAMPLE 1.  CHARACTER*5 A
            CALL SUBSTR('1234567890',A,4,5)
            PRINT,A
```

This will print '45678' since it starts at the 4th character of the string and moves 5 characters into A.

```
EXAMPLE 2.  CHARACTER*5 A/'12345'/
            CALL SUBSTR('ABCDEFGH',A,3,2,2)
```

In this case we have specified the displacement into the result as 2 so A is now '1CD45' since it takes 2 elements starting at the 3rd element of the string and puts it in the 2nd element of A.

```
EXAMPLE 3.  INTEGER*2    A(5)/1,2,3,4,5/,B(2)
            CALL SUBSTR(A,B,3,3)
```

Since B has only two elements it will be equal to 3 4 and the third element will be truncated.

```
EXAMPLE 4.  INTEGER A(5),B(2)
            CALL SUBSTR(A,B,4,5)
```

Since the field specified is longer than the source field (we are to take 5 elements starting at element 4 and the source is only 5 elements) we cancel execution with an error message which states that a length error has occurred. If the call had been

```
CALL SUBSTR(A,B,4,5,&3)
```

then instead of cancelling the run control would have been passed back to statement number 3 before any move is performed.

NOTE:

The RDISP parameter and RETURN parameter are completely independent and either one can be used alone or they can both be specified. However; if both are specified RDISP must be before RETURN or it will be ignored.

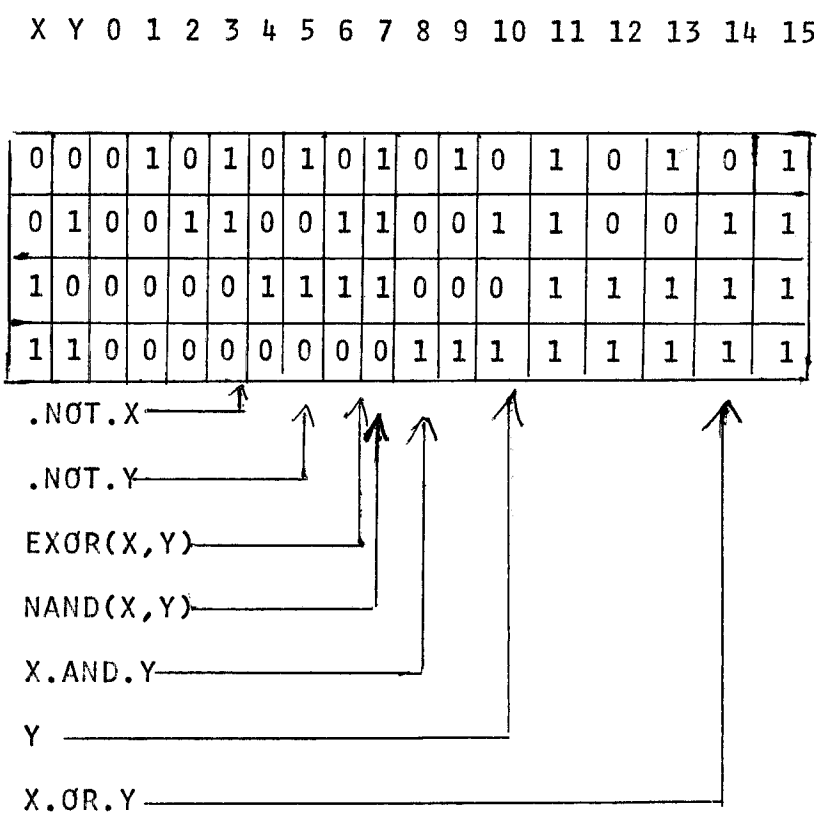
BOOL

This program will perform the logical boolean operation defined by the function parameter. Bool is a subroutine, therefore the result field must be supplied by the calling routine. The arguments, other than the function argument, may be any valid WATFIV data types as long as the total length in bytes is the same for all three arguments.

The calling sequence is:

```
CALL BOOL(ARG1,ARG2,RESULT,FUNC)
```

ARG1,ARG2, and RESULT can be any valid data types as long as the total length in bytes is equal,for example,they can be INTEGER*4,LOGICAL*4 or REAL*4. FUNC must be INTEGER*2 or INTEGER*4. Only the last four bits of FUNC are used and the logical operation it performs is defined by n1 n2 n3 n4, where n1,n2,n3,n4 are the low order four bits of FUNC. There are 16 possible bit combinations and so there are 16 possible boolean operations. The operations are defined as shown:


X Y 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1
1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
.NOT.X
.NOT.Y
EXOR(X,Y)
NAND(X,Y)
X.AND.Y
Y
X.OR.Y

EXAMPLE 1.

```
INTEGER*2 A(6)/1,3,5,7,9,2/
INTEGER*2 B(6)/2,4,6,8,0,1/,C
I=7
CALL BOOL(A(3),B(4),C,I)
```

In this example, the boolean operation equal to 7(0111) is performed on the integers 5(0101) and 8(1000). The result field expressed as an integer is equal to 13. To calculate the result, n1,n2,n3,n4 are replaced by 0,1,1,1 and the bits of the two arguments are set according to this operation.

EXAMPLE 2.

```
J=10
K=I=8
INTEGER*4 C
CALL BOOL(J,K,C,I)
```

In example 2, the boolean operation equal to 10(1010) is performed on the integers 10 and 8. In this case, after the boolean operation, the sign bit will be 1, so the result is negative. If we calculate the binary result and convert to decimal we find that the result is -11.