Reference Modification and Intrinsic Functions

Introduction

Aims

The aim of this unit is to provide to provide you with a solid understanding of string manipulation using Reference Modification. It will also introduce you to Intrinsic Functions and will show how strings may be manipulated using the the String Intrinsic Functions.

Objectives

By the end of this unit you should:

1. Understand how Reference Modification works
2. Be able to use Reference Modification to extract and insert sub-strings.
3. Understand how Intrinsic Functions work.
4. Be able to using Intrinsic Functions for string and date manipulation.

Prerequisites

You should be familiar with the material covered in the unit;

- Data Declaration
- Iteration
- Selection
- Tables
- Edited Pictures
- The INSPECT verb
- The STRING verb
- The UNSTRING verb

To top of page

Reference Modification

Definition & Syntax

Reference Modification allows you to treat a numeric (PIC 9) or alphanumeric (PIC X) data-item as if it were an array of characters. To access sub-strings using Reference Modification you must specify;

- the name of the data-item
- the start character position of the sub-string
- the number of characters in the sub-string

To apply Reference Modification the following syntax must be used.

DataName(*StartPos* **[:***SubStrLength***])**

StartPos is the character position of the first character in the sub-string and *SubStrLength* is number of characters in the substring.

As the square brackets indicate, the SubStrLength may be omitted, and in that case the substring from StartPos to the end of the string is assumed.

Reference Modification may be used almost anywhere an alphanumeric data-item is permitted.

Examples

```
WORKING-STORAGE SECTION.
01 xString    PIC X(38) VALUE "This is the alphanumeric string".
01 nString    PIC 9(5)V99 VALUE 34526.56.
01 SubStrSize PIC 99 VALUE 12.
01 StartPos   PIC 99 VALUE 18.
```

Intrinsic Functions

Introduction

Although COBOL does not permit user-defined functions or procedures, it does have a number of Intrinsic (built-in) Functions, which you can use in your programs.

These functions fall into three broad categories : date functions, numeric functions and string functions.

Using Intrinsic Functions

Like functions in other languages, an Intrinsic Function is replaced in the position where it occurs by the function result.

In COBOL, an Intrinsic Function is a temporary data item whose value is determined at the time the function is executed.

Whereever a literal with the same type as the function result may be referenced, the Intrinsic Function may be referenced.

Intrinsic Function Notes

- Where the function result is alphanumeric it has an implicit usage of DISPLAY.
- Functions that return a number value (numeric & integer) are always considered to be signed.
- A function that returns a number value can be used only in an arithmetic expression or as the source of a MOVE statement.
- A function that returns a numeric value cannot be used where an interger operand is required (because it may return a non-integer value).

Intrinsic Function Template

An Intrinsic Function consists of three parts;

1. The start of the function is signalled by the keyword - FUNCTION.
2. The keyword is followed by the name of the function.
3. The name of the function is immediately followed by the bracketed list of parameters or arguments.

The template for Intrinsic Functions is shown below:

FUNCTION FunctionName(*Parameters*)

FunctionName is the name of the function and *Parameters* is one or more parameters supplied to the function.

Example declarations.

```
MOVE FUNCTION RANDOM(99) TO RandNum.

DISPLAY FUNCTION UPPER-CASE("this will be in upper case").
```

Intrinsic Function Notation

In the Intrinsic Function definitions in the sections below the functions produce a result of one of the following types;

- Alphanumeric
- Numeric (includes Integer)
- Integer (does not allow the decimal point)

Where parameters are required in the function the parameter name is used to indicate the type of the parameter.

- **Alph** indicates Alphanumeric
- **Num** indicates any Numeric
- **PosNum** indicates a Positive Numeric
- **Int** indicates any Integer
- **PosInt** indicates a PositiveInteger
- **Any** indicates that any type may be used

Where a function takes a parameter list (indicated by **{Any}...** in the function definition) the parameter list may be replaced by an array. The reserved word ALL is used as the array subscript to indicate all the elements of the array.

For instance the ORD-MAX function may take a parameter list or an array may be used:

```
MOVE FUNCTION ORD-MAX(12 23 03 78 65) TO OrdPos
                      or
MOVE FUNCTION ORD-MAX(IntElement(ALL)) TO OrdPos
```

String Functions

Definitions

Function Name	Result Type	Comment
CHAR(PosInt)	Alphanumeric	Returns the character at ordinal position **PosInt** of the collating sequence.
ORD(Alph)	Integer	Returns the ordinal position of character **Alph**.
ORD-MAX({Any}...)	Integer	Returns the ordinal position of whichever of the parameters has the highest value. All parameters must be of the same type. The parameter list may be replaced by an array.
ORD-MIN({Any}...)	Integer	Returns the ordinal position of whichever of the parameters has the lowest value. All parameters must be of the same type.
LENGTH(Any)	Integer	Returns the number of characters in **Any**.
REVERSE(Alph)	Alphanumeric	Returns a character string with the characters in **Alph** reversed.
LOWER-CASE(Alph)	Alphanumeric	Returns a character string with the characters in **Alph** changed to their lower case equivalents.
UPPER-CASE(Alph)	Alphanumeric	Returns a character string with the characters in **Alph** changed to their upper case equivalents

Examples

The statements in the following examples produce the results shown below.

```
WORKING-STORAGE SECTION.
```

```
01 xString PIC X(38) VALUE "This is the alphanumeric string".
01 OrdPos PIC 99.
01 IntArray VALUE "1223037865".
   02 Ielement OCCURS 5 TIMES PIC 99.

PROCEDURE DIVISION.
Begin.
PROCEDURE DIVISION.
Begin.
  DISPLAY "Character at pos 39 is = " FUNCTION CHAR(39)

  MOVE FUNCTION ORD("A") TO OrdPos
  DISPLAY "Ord pos of A is = " OrdPos

  MOVE FUNCTION ORD-MAX("t" "b" "x" "s") TO OrdPos
  DISPLAY "Highest character in sequence is at pos " OrdPos

  MOVE FUNCTION ORD-MIN("t" "b" "x" "s") TO OrdPos
  DISPLAY "Lowest character in sequence is at pos " OrdPos

  MOVE FUNCTION ORD-MAX(Ielement(ALL)) TO OrdPos
  DISPLAY "Number " Ielement(OrdPos) " is the highest in array".

  DISPLAY "Length of xString is " FUNCTION LENGTH(xString)
  DISPLAY FUNCTION REVERSE(xString(1:31))
  DISPLAY FUNCTION UPPER-CASE(xString)
  DISPLAY FUNCTION LOWER-CASE(xString)
  STOP RUN.
```

Results	
Display 1 =	Character at pos 39 is = **&**
Display 2 =	Ord pos of A is = **66**
Display 3 =	Highest character in sequence is at pos **03**
Display 4 =	Lowest character in sequence is at pos **02**
Display 5 =	Number **78** is the highest in array
Display 6 =	Length of xString is **38**
Display 7 =	**gnirts ciremunahpla eht si sihT**
Display 8 =	**THIS IS THE ALPHANUMERIC STRING**
Display 9 =	**this is the alphanumeric string**

Date Functions

Definitions

Function Name	Result Type	Comment
CURRENT-DATE	Alphanumeric	Returns a 21 character string representing the current date and time, and the difference between the local time and Greenwich Mean Time. The format of the string is yyyymmddhhmmsshhxhhmm, where xhhmm is the number of hours and minutes the local time is ahead or behind GMT (x = + or - or 0). If x = 0, the hardware cannot provide this information.
DATE-OF-INTEGER(PosInt)	Integer	Returns the yyyymmdd (standard date) equivalent of the integer date - **PosInt**. The integer date is the number of days that have passed since Dec 31st 1600 in the Gregorian Calendar.

DAY-OF-INTEGER(PosInt)	Integer	Returns the yyyddd (Julien Date) equivalent of the integer date - **PosInt**.
INTEGER-OF-DATE(PosInt)	Integer	Returns the integer date equivalent of standard date **PosInt**. Posint is an integer of the form yyyymmdd.
INTEGER-OF-DAY(PosInt)	Integer	Returns the integer date equivalent of Julian date (yyyyddd) represented by **PosInt**.
WHEN-COMPILED	Integer	Returns the date and time the program was compiled. Uses the same format as CURRENT-DATE.

Examples

The example program below uses most of the date functions described in the table above. The results from running the program are shown below.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. DateFunctions.
AUTHOR. Michael Coughlan.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DateAndTimeC.
   02 DateC.
      03 YearC    PIC 9(4).
      03 MonthC   PIC 99.
      03 DayC     PIC 99.
   02 TimeC.
      03 HourC    PIC 99.
      03 MinC     PIC 99.
      03 SecC     PIC 99.
      03 HundredC PIC 99.
   02 GMT.
      03 GMTDiff  PIC X.
         88 GMTNotSupported VALUE "0".
      03 GMTHours PIC 99.
      03 GMTMins  PIC 99.
01 BillDate       PIC 9(8).
01 DateNow        PIC 9(8).
01 DaysOverdue    PIC S999.
01 NumOfDays      PIC 999.
01 IntFutureDate  PIC 9(8).
01 FutureDate     PIC 9(8).
01 DisplayDate REDEFINES FutureDate.
   02 YearD       PIC 9999.
   02 MonthD      PIC 99.
   02 DayD        PIC 99.

PROCEDURE DIVISION.
Begin.
* This example gets the current date and displays
* its constituent parts.
  MOVE FUNCTION CURRENT-DATE TO DateAndTimeC
  DISPLAY "Current Date is " MonthC "/" DayC "/" YearC
  DISPLAY "Current Time is " HourC ":" MinC ":" SecC
  IF GMTNotSupported
     DISPLAY "This computer cannot supply the time"
     DISPLAY "difference between local and GMT."
   ELSE
     DISPLAY "The local time is - GMT "
              GMTDiff GMTHours ":" GMTMins
  END-IF.

* In this example bills fall due 30 days from
* the billing date.
```

```
           DISPLAY "Enter the date of the bill (yyyymmdd) "
                    WITH NO ADVANCING
           ACCEPT BillDate
           MOVE DateC TO DateNow
           COMPUTE DaysOverDue = (FUNCTION INTEGER-OF-DATE(DateNow))
                               - (FUNCTION INTEGER-OF-DATE(BillDate)+ 30)
           EVALUATE TRUE
             WHEN DaysOverDue > ZERO DISPLAY "This bill is over due"
             WHEN DaysOverDue = ZERO DISPLAY "This bill is due today"
             WHEN DaysOverDue < ZERO DISPLAY "This bill is not yet due"
           END-EVALUATE

         * This example displays the date NumOfDays days
         * from the current date
           DISPLAY "Enter the number of days - "
           WITH NO ADVANCING
           ACCEPT NumOfDays
           COMPUTE IntFutureDate =
           FUNCTION INTEGER-OF-DATE(DateNow)+ NumOfDays + 1
           MOVE FUNCTION DATE-OF-INTEGER(IntFutureDate) TO FutureDate
           DISPLAY "The date in "
           NumOfDays " days time will be " MonthD "/" DayD "/" YearD
           STOP RUN.
```

Results

```
Current Date is 01/26/1998
Current Time is 12:20:17
The local time is - GMT +00:00
Enter the date of the bill (yyyymmdd) 19971227
This bill is due today
Enter the number of days - 365
The date in 365 days time will be 01/27/1999
```

Miscellaneous Functions

Other Functions

Function Name	Result Type	Comment
ACOS(Num)	Numeric	Returns a numeric value in radians that approximates the arccosine of **Num**. The value of Num must be greater than or equal to −1 and less than or equal to +1.
ANNUITY(Num PosInt)	Numeric	Returns a numeric value approximating the ratio of an annuity paid at the end of each period for the number of periods specified by **PosInt** to an initial investment of one. Interest is earned at the rate specified by **Num** and is applied at the end of the period, before the payment.
ASIN(Num)	Numeric	Returns a numeric value in radians that approximates the arcsine of **Num**.
ATAN(Num)	Numeric	Returns a numeric value in radians that approximates the arctangent of

		Num.
FACTORIAL(PosInt)	Integer	Returns an integer that is the factorial of **PosInt**.
INTEGER(Num)	Integer	Returns the greatest integer value that is less than or equal to **Num**. For instance -2 is returned if the Num has a value of -1.5 and 1 is returned if it has a value of 1.5.
INTEGER-PART(Num)	Integer	Returns the integer part of **Num** so a value of -1.5 is returned as -1 and a value of 1.5 is returned as 1.
LOG(PosNum)	Numeric	Returns a numeric value that approximates the logarithm to the base e (natural log) of **PosNum**. PosNum must be greater than 0.
LOG10(PosNum)	Numeric	Returns a numeric value that approximates the logarithm to the base 10 of **PosNum**. PosNum must be greater than 0.
MAX({Any}...)	Depends on type of **Any** see note.	Takes a parameter list and returns the content of whichever parameter contains the maximum value. Note. The returned type depends upon the parameter types as follows; **Alphanumeric** if parameters are Alphabetic or Alphnumeric. **Integer** if all are integer. **Numeric** if all are Numeric or mixed with Integer. An array may be used instead of the parameter list.
MEAN({Num}...)	Numeric	Returns a numeric value that is the arithmetic mean (average) of its parameters. An array may be used.
MEDIAN({Num}...)	Numeric	Returns the middle value of a list of values after the values have been arranged in sorted order. An array may be used.
MIDRANGE({Num}...)	Numeric	Returns a numeric value that is the arithmetic mean (average) of the minimum value and the maximum value. An array may be used.
MIN({Any}...)	Depends on type of **Any** see note in MAX above.	Takes a parameter list and returns the content of whichever parameter contains the minimum value. An array may be used.
MOD(Int1 Int2)	Integer	Returns an integer value defined as **Int1 – (Int2** * FUNCTION INTEGER (Int1 / Int2)).
NUMVAL(Alph)	Numeric	Converts the edited numeric string contained in **Alph** to a numeric item.

		Leading and trailing spaces are ignored and + - CR DB and the decimal point are stripped off.
PRESENT-VALUE(Num1 {Num2}...)	Numeric	Returns the amount to be invested today to produce a future value at a particular rate of interest. **Num1** is the percent interest rate expressed as a decimal value (.7 = 7%). **Num2** is the future desired value at the end of the period.
RANDOM(PosInt) or RANDOM	Numeric	Returns a numeric value that is a pseudo-random number. If a parameter is used then **PosInt** is the seed. Subsequent references without the parameter return the next number in the sequence. A value >0 and <1 will be returned.
RANGE({Num1}...)	Integer if all parameter are Integer else Numeric	Examines a list of parameters and returns a value that is equal to the value of the maximum argument minus the value of the minimum argument.
REM(Num1 Num2)	Numeric	Returns a numeric value that is the remainder of **Num1** divided by **Num2**.
SIN(Num)	Numeric	Returns an approximation of the sine of an angle or arc, expressed in radians, that is specified by **Num**.
SQRT(Num)	Numeric	Returns an approximation of the square root of **Num**.
STANDARD-DEVIATION({Num}...)	Numeric	Returns an approximation of the standard deviation of its parameters.
SUM({Num}...)	Integer if all parameter are Integer else Numeric	Returns the sum of the parameters.
TAN(Num)	Numeric	Returns an approximation of the tangent of an angle or arc, expressed in radians, that is specified by **Num**.
VARIANCE({Num}...)	Numeric	Returns an approximation of the variance of its arguments

General Examples

The example program below uses many of the functions described in the table above. The results from running the program are shown below.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  OtherFunctions.
AUTHOR.  Michael Coughlan.
* An example program using misc Intrinsic Functions

DATA DIVISION.
WORKING-STORAGE SECTION.
01  IntParameters.
```

```
    02 IP1          PIC 99   VALUE 25.
    02 IP2          PIC 99   VALUE 50.
    02 IP3          PIC 9(5) VALUE 12.
    02 IResult      PIC 9(5).

01  NumResult       PIC 9.9(5).

01 IntArray VALUE "1223037865".
   02 Ielement OCCURS 5 TIMES PIC 99.

PROCEDURE DIVISION.
Begin.
  MOVE FUNCTION MAX(IP1 545 IP3 IP2) TO IResult
  DISPLAY "Max value is = " IResult

  MOVE FUNCTION MAX(Ielement(ALL)) TO IResult
  DISPLAY "Max value is = " IResult

  MOVE FUNCTION MEDIAN(Ielement(ALL)) TO IResult
  DISPLAY "Median value is = " IResult

  MOVE FUNCTION MIDRANGE(Ielement(ALL)) TO IResult
  DISPLAY "Midrange value is = " IResult

  MOVE FUNCTION MIN(Ielement(ALL)) TO IResult
  DISPLAY "Min value is = " IResult

  MOVE FUNCTION RANGE(Ielement(ALL)) TO IResult
  DISPLAY "Range is " IResult

  MOVE FUNCTION SUM(Ielement(ALL)) TO IResult
  DISPLAY "The sum of the values is " IResult

  MOVE FUNCTION VARIANCE(Ielement(ALL)) TO IResult
  DISPLAY "The variance of the values is " IResult
  STOP RUN.
```

Results

```
Max value is = 00545
Max value is = 00078
Median value is = 00023
Midrange value is = 00040
Min value is = 00003
The sum of the values is 00181
The variance of the values is 00887
```

Lotto Example

The example program below uses the RANDOM function to get and display 6 unique lotto numbers. Valid lotto numbers fall between 1 and 42.

In this program the last five digits of the system time (i.e. minutes, seconds and hundreths of seconds) is used as the random number seed.

The algorithm is based on Niklaus Wirth's algorithm for finding a set of unique integers. In this algorithm when the loop test in executed -

> i + 1 is the position where the current number was inserted
> j is the first position where the current number was found
> If i + 1 = j then the current number has no duplicates.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  Lotto.
AUTHOR.  Michael Coughlan.
* This program displays six unique random lotto numbers.
```

```
* It was Adapted from a program written by Dermot Shinners-Kennedy

DATA DIVISION.
WORKING-STORAGE SECTION.
01 LottoNum  PIC 9(5) OCCURS 6 TIMES INDEXED BY j.
01 i         PIC 9(3).
01 RandNum   PIC 9(3).
01 SysDate.
   02 FILLER PIC 9(11).
   02 Seed   PIC 9(5).

PROCEDURE DIVISION.
Begin.
   MOVE FUNCTION CURRENT-DATE TO SysDate
   COMPUTE RandNum = FUNCTION RANDOM(Seed)

   SET i to ZERO
   PERFORM 6 TIMES
      PERFORM WITH TEST AFTER UNTIL i + 1 = j
        COMPUTE LottoNum(i + 1), RandNum =
                (FUNCTION RANDOM * 42) + 1
        SET j to 1
        SEARCH LottoNum
           WHEN LottoNum(j) = RandNum CONTINUE
        END-SEARCH
      END-PERFORM
      ADD 1 to i
      DISPLAY LottoNum(i) SPACE WITH NO ADVANCING
   END-PERFORM
   STOP RUN.
```

To top of page

Copyright Notice

Last updated : August 1998
e-mail : CSISwebeditor@ul.ie