COBOL - BASIC VERBS

COBOL verbs are used in the procedure division for data processing. A statement always start with a COBOL verb. There are several COBOL verbs with different types of actions.

Input / Output Verbs

Input/Output verbs are used to get data from the user and display the output of COBOL programs. The following two verbs are used for this process:

Accept Verb

Accept verb is used to get data such as date, time, and day from the operating system or directly the from user. If a program is accepting data from the user, then it needs to be passed through JCL. While getting data from the operating system FROM option is included as shown in the following below example:

```
ACCEPT WS-STUDENT-NAME.
ACCEPT WS-DATE FROM SYSTEM-DATE.
```

Display Verb

Display verb is used to display the output of a COBOL program.

```
DISPLAY WS-STUDENT-NAME.
DISPLAY "System date is : " WS-DATE.
```

COBOL PROGRAM

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-STUDENT-NAME PIC X(25).
   01 WS-DATE PIC X(10).

PROCEDURE DIVISION.
   ACCEPT WS-STUDENT_NAME.
   ACCEPT WS-DATE FROM DATE.
   DISPLAY "Name :  " WS-STUDENT_NAME.
   DISPLAY "Date : " WS-DATE.

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
//INPUT DD DSN=PROGRAM.DIRECTORY,DISP=SHR
//SYSIN DD *
TutorialsPoint
/*
```

When you compile and execute the above program, it produces the following result:

```
Name : TutorialsPoint
Date : 2014-08-30
```

Initialize Verb

Initialize verb is used to initialize a group item or an elementary item. Data names with RENAME clause cannot be initialized. Numeric data items are replaced by ZEROES. Alphanumeric or alphabetic data items are replaced by SPACES. If we include REPLACING term, then data items can be initialized to the given replacing value as shown in the following example:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NAME PIC A(30) VALUE 'ABCDEF'.
   01 WS-ID PIC 9(5).
   01 WS-ADDRESS.
   05 WS-HOUSE-NUMBER PIC 9(3).
   05 WS-COUNTRY PIC X(15).
   05 WS-PINCODE PIC 9(6) VALUE 123456.

PROCEDURE DIVISION.
   A000-FIRST-PARA.
   INITIALIZE WS-NAME, WS-ADDRESS.
   INITIALIZE WS-ID REPLACING NUMERIC DATA BY 12345.
   DISPLAY "My name is    : "WS-NAME.
   DISPLAY "My ID is      : "WS-ID.
   DISPLAY "Address       : "WS-ADDRESS.
   DISPLAY "House Number  : "WS-HOUSE-NUMBER.
   DISPLAY "Country       : "WS-COUNTRY.
   DISPLAY "Pincode       : "WS-PINCODE.

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
My name is    :
My ID is      : 12345
Address       : 000               000000
House Number  : 000
Country       :
Pincode       : 000000
```

Move Verb

Move verb is used to copy data from source data to destination data. It can be used on both elementary and group data items. For group data items, MOVE CORRESPONDING/CORR is used. In try it option, MOVE CORR is not working; but on a mainframe server it will work.

For moving data from a string, MOVE$x:l$ is used where x is the starting position and l is the length. Data will be truncated if destination data item PIC clause is less than the source data item PIC clause. If the destination data item PIC clause is more than the source data item PIC clause, then ZEROS or SPACES will be added in the extra bytes. The following example makes it clear:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9).
   01 WS-NUM2 PIC 9(9).
   01 WS-NUM3 PIC 9(5).
   01 WS-NUM4 PIC 9(6).
   01 WS-ADDRESS.
   05 WS-HOUSE-NUMBER PIC 9(3).
```

```
   05 WS-COUNTRY PIC X(5).
   05 WS-PINCODE PIC 9(6).
   01 WS-ADDRESS1.
   05 WS-HOUSE-NUMBER1 PIC 9(3).
   05 WS-COUNTRY1 PIC X(5).
   05 WS-PINCODE1 PIC 9(6).

PROCEDURE DIVISION.
   A000-FIRST-PARA.
   MOVE 123456789 TO WS-NUM1.
   MOVE WS-NUM1 TO WS-NUM2 WS-NUM3.
   MOVE WS-NUM1(3:6) TO WS-NUM4.
   MOVE 123 TO WS-HOUSE-NUMBER.
   MOVE 'INDIA' TO WS-COUNTRY.
   MOVE 112233 TO WS-PINCODE.
   MOVE WS-ADDRESS TO WS-ADDRESS1.

   DISPLAY "WS-NUM1     : " WS-NUM1
   DISPLAY "WS-NUM2     : " WS-NUM2
   DISPLAY "WS-NUM3     : " WS-NUM3
   DISPLAY "WS-NUM4     : " WS-NUM4
   DISPLAY "WS-ADDRESS  : " WS-ADDRESS
   DISPLAY "WS-ADDRESS1 : " WS-ADDRESS1

STOP RUN.
```

JCL to execute the above COBOL program.

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program it produces the following result:

```
WS-NUM1     : 123456789
WS-NUM2     : 123456789
WS-NUM3     : 56789
WS-NUM4     : 345678
WS-ADDRESS  : 123INDIA112233
WS-ADDRESS1 : 123INDIA112233
```

Legal Moves

The following table gives information about the legal moves:

	Alphabetic	Alphanumeric	Numeric
Alphabetic	Possible	Possible	Not Possible
Alphanumeric	Possible	Possible	Possible
Numeric	Not Possible	Possible	Possible

Add Verb

Add verb is used to add two or more numbers and store the result in the destination operand.

Syntax

give below is the syntax to Add two or more numbers:

```
ADD A B TO C D

ADD A B C TO D GIVING E
```

```
ADD CORR WS-GROUP1 TO WS-GROUP2
```

In syntax-1, A, B, C are added and the result is stored in C $C = A + B + C$. A, B, D are added and the result is stored in D $D = A + B + D$.

In syntax-2, A, B, C, D are added and the result is stored in E $E = A + B + C + D$.

In syntax-3, sub-group items with in WS-GROUP1 and WS GROUP2 are the added and result is stored in WS-GROUP2.

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9) VALUE 10 .
   01 WS-NUM2 PIC 9(9) VALUE 10.
   01 WS-NUM3 PIC 9(9) VALUE 10.
   01 WS-NUM4 PIC 9(9) VALUE 10.
   01 WS-NUMA PIC 9(9) VALUE 10.
   01 WS-NUMB PIC 9(9) VALUE 10.
   01 WS-NUMC PIC 9(9) VALUE 10.
   01 WS-NUMD PIC 9(9) VALUE 10.
   01 WS-NUME PIC 9(9) VALUE 10.

PROCEDURE DIVISION.
   ADD WS-NUM1 WS-NUM2 TO WS-NUM3 WS-NUM4.
   ADD WS-NUMA WS-NUMB WS-NUMC TO WS-NUMD GIVING WS-NUME.
   DISPLAY "WS-NUM1     : " WS-NUM1
   DISPLAY "WS-NUM2     : " WS-NUM2
   DISPLAY "WS-NUM3     : " WS-NUM3
   DISPLAY "WS-NUM4     : " WS-NUM4
   DISPLAY "WS-NUMA     : " WS-NUMA
   DISPLAY "WS-NUMB     : " WS-NUMB
   DISPLAY "WS-NUMC     : " WS-NUMC
   DISPLAY "WS-NUMD     : " WS-NUMD
   DISPLAY "WS-NUME     : " WS-NUME

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
WS-NUM1     : 000000010
WS-NUM2     : 000000010
WS-NUM3     : 000000030
WS-NUM4     : 000000030
WS-NUMA     : 000000010
WS-NUMB     : 000000010
WS-NUMC     : 000000010
WS-NUMD     : 000000010
WS-NUME     : 000000040
```

Subtract Verb

Subtract verb is used for subtraction operations.

Syntax

given below is the syntax for Subtract operations:

```
SUBTRACT A B FROM C D

SUBTRACT A B C FROM D GIVING E

SUBTRACT CORR WS-GROUP1 TO WS-GROUP2
```

In syntax-1, A and B are added and subtracted from C. The Result is stored in C $C = C - (A + B)$. A and B are added and subtracted from D. The result is stored in D $D = D - (A + B)$.

In syntax-2, A, B, C are added and subtracted from D. Result is stored in E $E = D - (A + B + C)$

In syntax-3, sub-group items within WS-GROUP1 and WS-GROUP2 are subtracted and the result is stored in WS-GROUP2.

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9) VALUE 10 .
   01 WS-NUM2 PIC 9(9) VALUE 10.
   01 WS-NUM3 PIC 9(9) VALUE 100.
   01 WS-NUM4 PIC 9(9) VALUE 100.
   01 WS-NUMA PIC 9(9) VALUE 10.
   01 WS-NUMB PIC 9(9) VALUE 10.
   01 WS-NUMC PIC 9(9) VALUE 10.
   01 WS-NUMD PIC 9(9) VALUE 100.
   01 WS-NUME PIC 9(9) VALUE 10.

PROCEDURE DIVISION.
   SUBTRACT WS-NUM1 WS-NUM2 FROM WS-NUM3 WS-NUM4.
   SUBTRACT WS-NUMA WS-NUMB WS-NUMC FROM WS-NUMD GIVING WS-NUME.

   DISPLAY "WS-NUM1     : " WS-NUM1
   DISPLAY "WS-NUM2     : " WS-NUM2
   DISPLAY "WS-NUM3     : " WS-NUM3
   DISPLAY "WS-NUM4     : " WS-NUM4
   DISPLAY "WS-NUMA     : " WS-NUMA
   DISPLAY "WS-NUMB     : " WS-NUMB
   DISPLAY "WS-NUMC     : " WS-NUMC
   DISPLAY "WS-NUMD     : " WS-NUMD
   DISPLAY "WS-NUME     : " WS-NUME

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
WS-NUM1     : 000000010
WS-NUM2     : 000000010
WS-NUM3     : 000000080
WS-NUM4     : 000000080
WS-NUMA     : 000000010
WS-NUMB     : 000000010
WS-NUMC     : 000000010
WS-NUMD     : 000000100
WS-NUME     : 000000070
```

Multiply Verb

Multiply verb is used for multiplication operations.

Syntax

Given below is the syntax to multiply two or more numbers:

```
MULTIPLY A BY B C

MULTIPLY A BY B GIVING E
```

In syntax-1, A and B are multipled and the result is stored in B $B = A * B$. A and C are multipled and the result is stored in C $C = A * C$.

In syntax-2, A and B are multipled and the result is stored in E $E = A * B$.

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9) VALUE 10 .
   01 WS-NUM2 PIC 9(9) VALUE 10.
   01 WS-NUM3 PIC 9(9) VALUE 10.
   01 WS-NUMA PIC 9(9) VALUE 10.
   01 WS-NUMB PIC 9(9) VALUE 10.
   01 WS-NUMC PIC 9(9) VALUE 10.

PROCEDURE DIVISION.
   MULTIPLY WS-NUM1 BY WS-NUM2 WS-NUM3.
   MULTIPLY WS-NUMA BY WS-NUMB GIVING WS-NUMC.

   DISPLAY "WS-NUM1     : " WS-NUM1
   DISPLAY "WS-NUM2     : " WS-NUM2
   DISPLAY "WS-NUM3     : " WS-NUM3
   DISPLAY "WS-NUMA     : " WS-NUMA
   DISPLAY "WS-NUMB     : " WS-NUMB
   DISPLAY "WS-NUMC     : " WS-NUMC

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
WS-NUM1     : 000000010
WS-NUM2     : 000000100
WS-NUM3     : 000000100
WS-NUMA     : 000000010
WS-NUMB     : 000000010
WS-NUMC     : 000000100
```

Divide Verb

Divide verb is used for division operations.

Syntax

Given below following is the syntax for division operations:

```
DIVIDE A INTO B
```

```
DIVIDE A BY B GIVING C REMAINDER R
```

In syntax-1, B is divided by A and the result is stored in B $B = B/A$.

In syntax-2, A is divided by B and the result is stored in C $C = A/B$ and remainder is stored in R.

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9) VALUE 5.
   01 WS-NUM2 PIC 9(9) VALUE 250.
   01 WS-NUMA PIC 9(9) VALUE 100.
   01 WS-NUMB PIC 9(9) VALUE 15.
   01 WS-NUMC PIC 9(9).
   01 WS-REM PIC 9(9).

PROCEDURE DIVISION.
   DIVIDE WS-NUM1 INTO WS-NUM2.
   DIVIDE WS-NUMA BY WS-NUMB GIVING WS-NUMC REMAINDER WS-REM.
   DISPLAY "WS-NUM1     : " WS-NUM1
   DISPLAY "WS-NUM2     : " WS-NUM2
   DISPLAY "WS-NUMA     : " WS-NUMA
   DISPLAY "WS-NUMB     : " WS-NUMB
   DISPLAY "WS-NUMC     : " WS-NUMC
   DISPLAY "WS-REM      : " WS-REM

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
WS-NUM1     : 000000005
WS-NUM2     : 000000050
WS-NUMA     : 000000100
WS-NUMB     : 000000015
WS-NUMC     : 000000006
WS-REM      : 000000010
```

Compute Statement

Compute statement is used to write arithmetic expressions in COBOL. This is a replacement for Add, Subtract, Multiply, and Divide.

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9) VALUE 10 .
   01 WS-NUM2 PIC 9(9) VALUE 10.
   01 WS-NUM3 PIC 9(9) VALUE 10.
   01 WS-NUMA PIC 9(9) VALUE 50.
   01 WS-NUMB PIC 9(9) VALUE 10.
   01 WS-NUMC PIC 9(9).

PROCEDURE DIVISION.
```

```
   COMPUTE WS-NUMC= (WS-NUM1 * WS-NUM2) - (WS-NUMA / WS-NUMB) + WS-NUM3.
   DISPLAY "WS-NUM1      : " WS-NUM1
   DISPLAY "WS-NUM2      : " WS-NUM2
   DISPLAY "WS-NUM3      : " WS-NUM3
   DISPLAY "WS-NUMA      : " WS-NUMA
   DISPLAY "WS-NUMB      : " WS-NUMB
   DISPLAY "WS-NUMC      : " WS-NUMC

STOP RUN.
```

JCL to execute the above COBOL program.

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
WS-NUM1      : 000000010
WS-NUM2      : 000000010
WS-NUM3      : 000000010
WS-NUMA      : 000000050
WS-NUMB      : 000000010
WS-NUMC      : 000000105
```