COBOL - SUBROUTINES

Cobol subroutine is a program that can be compiled independently but cannot be executed independently. There are two types of subroutines: **internal subroutines** like **Perform** statements and **external subroutines** like **CALL** verb.

Call Verb

Call verb is used to transfer the control from one program to another program. The program that contains the CALL verb is the **Calling Program** and the program being called is known as the **Called Program**. Calling program execution will halt until the called program finishes the execution. Exit Program statement is used in the Called program to transfer the control back.

Called Program Constraints

Following are the called program requirements:

- **Linkage section** must be defined in the called program. It consists of data elements passed in the program. The data items should not have Value clause. PIC clause must be compatible with the variables passed through the calling program.
- **Procedure division using** has a list of variables passed from the calling program and the order must be same as mentioned in the Call verb.
- **Exit program** statement is used in the Called program to transfer the control back. It must be the last statement in the called program.

The parameters can be passed between programs in two ways:

- By Reference
- By Content

Call By Reference

If the values of variables in the called program are modified, then their new values will reflect in the calling program. If **BY** clause is not specified, then variables are always passed by reference.

Syntax

Following is the syntax of calling subroutine by reference:

```
CALL sub-prog-name USING variable-1, variable-2.
```

Example

Following **example** is the MAIN calling program and UTIL is the called program:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-STUDENT-ID PIC 9(4) VALUE 1000.
   01 WS-STUDENT-NAME PIC A(15) VALUE 'Tim'.

PROCEDURE DIVISION.
   CALL 'UTIL' USING WS-STUDENT-ID, WS-STUDENT-NAME.
   DISPLAY 'Student Id : ' WS-STUDENT-ID
   DISPLAY 'Student Name : ' WS-STUDENT-NAME
STOP RUN.
```

Called Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID. UTIL.

DATA DIVISION.
   LINKAGE SECTION.
   01 LS-STUDENT-ID PIC 9(4).
   01 LS-STUDENT-NAME PIC A(15).

PROCEDURE DIVISION USING LS-STUDENT-ID, LS-STUDENT-NAME.
   DISPLAY 'In Called Program'.
   MOVE 1111 TO LS-STUDENT-ID.
EXIT PROGRAM.
```

JCL to execute the above COBOL program:

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

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

```
In Called Program
Student Id : 1111
Student Name : Tim
```

Call By Content

If the values of variables in the called program are modified, then their new values will not reflect in the calling program.

Syntax

Following is the syntax of calling subroutine by content:

```
CALL sub-prog-name USING
BY CONTENT variable-1, BY CONTENT variable-2.
```

Example

Following **example** is the MAIN calling program and UTIL is the called program:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-STUDENT-ID PIC 9(4) VALUE 1000.
   01 WS-STUDENT-NAME PIC A(15) VALUE 'Tim'.

PROCEDURE DIVISION.
   CALL 'UTIL' USING BY CONTENT WS-STUDENT-ID, BY CONTENT WS-STUDENT-NAME.
   DISPLAY 'Student Id : ' WS-STUDENT-ID
   DISPLAY 'Student Name : ' WS-STUDENT-NAME
STOP RUN.
```

Called Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID. UTIL.

DATA DIVISION.
   LINKAGE SECTION.
```

```
   01 LS-STUDENT-ID PIC 9(4).
   01 LS-STUDENT-NAME PIC A(15).

PROCEDURE DIVISION USING LS-STUDENT-ID, LS-STUDENT-NAME.
   DISPLAY 'In Called Program'.
   MOVE 1111 TO LS-STUDENT-ID.
EXIT PROGRAM.
```

JCL to execute the above COBOL program:

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

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

```
In Called Program
Student Id : 1000
Student Name : Tim
```

Types of Call

There are two types of calls:

- **Static Call** occurs when a program is compiled with the NODYNAM compiler option. A static called program is loaded into storage at compile time.
- **Dynamic Call** occurs when a program is compiled with the DYNAM and NODLL compiler option. A dynamic called program is loaded into storage at runtime.