Capgemini

-Subroutines – Additional Reading

Lesson Objectives

In this lesson, participants will learn

- Subroutines

© 2018 Capgemini. All rights reserved.

Leaving a Subroutine

You can exit of a subroutine at any time using the following statements:

- exit
- check
- Stop

In subroutines

- check and exit immediately leave the subroutine and processing continues with the next executable statement following the perform
- stop immediately leaves the subroutine and goes directly to the end-of-selection event

© 2018 Capgemini. All rights reserved.

Leaving subroutine using Exit - Example

```
REPORT  Z.
DATA F1 VALUE 'X'.
PERFORM S1.
WRITE: / 'After invoking s1'.

FORM S1.
  WRITE / 'In s1'.
  EXIT.
  WRITE / 'After Exit in S1'.
ENDFORM.

*The exit statement in the subroutine exits the
*subroutine and  control goes to the statement
*after "Perform s1"
```

© 2018 Capgemini. All rights reserved.

Demo: Leaving subroutine using Exit

```
1   REPORT  Z.
2   DATA F1 VALUE 'X'.
3   PERFORM S1.
4   WRITE: / 'After invoking s1'.
5
6   FORM S1.
7     WRITE / 'In s1'.
8     EXIT.
9     WRITE / 'After Exit in S1'.
10  ENDFORM.
11
12  *The exit statement in the subroutine exits the
13  *subroutine and  control goes to the statement
14  *after "Perform s1"
```

Output after execution

```
In s1
After invoking s1
```

© 2018 Capgemini. All rights reserved.

Leaving subroutine using Check - Example

```
REPORT  Z.
DATA F1 VALUE 'X'.
PERFORM S2.
WRITE: / 'After invoking S2'.

FORM S2.
  WRITE / 'In S2'.
  CHECK F1 = 'Y'.
  WRITE / 'After Check in S2'.
ENDFORM.
```

**In form s2, when the statement check F1 = 'Y', is encountered*
**control comes out of the subroutine, since the check statement leaves the subroutine.*
**Use check to terminate a subroutine conditionally.*
**The subroutine is terminated if the logical expression in the CHECK statement IS untrue,*
**and the calling program resumes processing after the PERFORM statement.*

© 2018 Capgemini. All rights reserved.

Demo: Leaving subroutine using Check

Logical expression in the CHECK statement is untrue

```
1   REPORT  Z.
2   DATA F1 VALUE 'X'.
3   PERFORM S2.
4   WRITE: / 'After invoking S2'.
5
6   FORM S2.
7     WRITE / 'In S2'.
8     CHECK F1 = 'Y'.
9     WRITE / 'After Check in S2'.
10  ENDFORM.
```

- *If the logical expression in the CHECK statement IS untrue,*
- *the subroutine is terminated*

Output after execution

```
In S2
After invoking S2
```

© 2018 Capgemini. All rights reserved.

Demo: Leaving subroutine using Check

Logical expression in the CHECK statement is true

```
1   REPORT  Z.
2   DATA F1 VALUE 'Y'.
3   PERFORM S2.
4   WRITE: / 'After invoking S2'.
5
6   FORM S2.
7     WRITE / 'In S2'.
8     CHECK F1 = 'Y'.
9     WRITE / 'After Check in S2'.
10  ENDFORM.
```

- *If the logical expression in the CHECK statement IS true,*
- *the subroutine is NOT terminated*

Output after execution

```
In S2
After Check in S2
After invoking S2
```

© 2018 Capgemini. All rights reserved.

Leaving subroutine using Stop - Example

```
REPORT Z.
DATA F1 VALUE 'X'.
PERFORM S1.
WRITE: / 'After invoking S1'.
*
END-OF-SELECTION.
  WRITE:  'In end of selection'.

FORM S1.
 WRITE: / 'In S1'.
 STOP.
 WRITE / 'After Stop in S1'.
ENDFORM.
```

**A stop statement within the subroutine transfers control directly to end-of-selection.*

© 2018 Capgemini. All rights reserved.

Demo: Leaving subroutine using stop

```
1   REPORT Z.
2   DATA F1 VALUE 'X'.
3   PERFORM S1.
4   WRITE: / 'After invoking S1'.
5   *
6   END-OF-SELECTION.
7     WRITE:  'In end of selection'.
8
9   FORM S1.
10   WRITE: / 'In S1'.
11   STOP.
12   WRITE / 'After Stop in S1'.
13  ENDFORM.
14  *A stop statement within the subroutine
15  *transfers control directly to end-of-selection.
```

Output after execution

```
In S1
In end of selection
```

© 2018 Capgemini. All rights reserved.

Passing Parameters by Value and Result

Pass by value and result is very similar to pass by value.

Like pass by value, a new memory area is allocated and it holds an independent copy of the variable.

It is freed when the subroutine ends, and that is also when the difference occurs.

When the endform statement executes, it copies the value of the local memory area back into the original memory area.

Changes to the parameter within the subroutine are reflected in the original, but not until the subroutine returns

This may seem like a small difference, but the difference becomes greater.

You can change whether the copy takes place or not.

The copy always takes place unless you leave the subroutine by using one of the two statements:

- Stop
- Message ennn(error message)

© 2018 Capgemini. All rights reserved.

Passing Parameters by Value and Result - Example

```
REPORT Z.
DATA: F1 TYPE C VALUE 'A'.
WRITE: / 'Before invoking S2, F1 =', F1.
*PERFORM S2 USING F1.
PERFORM S2 CHANGING F1.
WRITE: / 'After invoking  S2, F1 =', F1.

END-OF-SELECTION.
WRITE: / 'In EOS, After invoking S2, F1 =', F1.

FORM S2 CHANGING VALUE(P1).
*Above is pass by value and result
P1 = 'X'.
WRITE: / 'In Subroutine S2, P1 =', P1.
*stop statement terminates the subroutine and
"Control goes to the end-of-selection event
STOP.
ENDFORM.
```

© 2018 Capgemini. All rights reserved.

Demo: Subroutine – Pass By Value and Result

Code – with stop in subroutine

```
1   REPORT Z.
2   DATA: F1 TYPE C VALUE 'A'.
3   WRITE: / 'Before invoking S2, F1 =', F1.
4   PERFORM S2 CHANGING F1.
5   WRITE: / 'After invoking  S2, F1 =', F1.
6   END-OF-SELECTION.
7   WRITE: / 'In EOS, After invoking S2, F1 =', F1.
8   FORM S2 CHANGING VALUE(P1).
9   *Above is pass by value and result
10  P1 = 'X'.
11  WRITE: / 'In Subroutine S2, P1 =', P1.
12  *stop statement terminates the subroutine and
13  *Control goes to the end-of-selection event.
14  STOP.
15  ENDFORM.
```

Output after execution

```
Before invoking S2, F1 = A
In Subroutine S2, P1 = X
In EOS,After invoking S2, F1 = A
```

© 2018 Capgemini. All rights reserved.

Demo: Subroutine – Pass By Value and Result

Code – without stop in subroutine

```
1   REPORT Z.
2   DATA: F1 TYPE C VALUE 'A'.
3   WRITE: / 'Before invoking S2, F1 =', F1.
4   *PERFORM S2 USING F1.
5   PERFORM S2 CHANGING F1.
6   WRITE: / 'After invoking  S2, F1 =', F1.
7
8   END-OF-SELECTION.
9   WRITE: / 'In EOS,After invoking S2, F1 =', F1.
10
11  FORM S2 CHANGING VALUE(P1).
12  *Above is pass by value and result
13  P1 = 'X'.
14  WRITE: / 'In Subroutine S2, P1 =', P1.
15  ENDFORM.
```

Output after execution

```
Before invoking S2, F1 = A
In Subroutine S2, P1 = X
After invoking  S2, F1 = X
In EOS,After invoking S2, F1 = X
```

© 2018 Capgemini. All rights reserved.

Demo: Subroutine – Calling External subroutine

Code – with stop in subroutine

```
1   REPORT Z.
2   DATA: F1 TYPE C VALUE 'A'.
3   WRITE: / 'Before invoking S2, F1 =', F1.
4   PERFORM S2 CHANGING F1.
5   WRITE: / 'After invoking  S2, F1 =', F1.
6   END-OF-SELECTION.
7   WRITE: / 'In EOS, After invoking S2, F1 =', F1.
8   FORM S2 CHANGING VALUE(P1).
9   *Above is pass by value and result
10  P1 = 'X'.
11  WRITE: / 'In Subroutine S2, P1 =', P1.
12  *stop statement terminates the subroutine and
13  *Control goes to the end-of-selection event.
14  STOP.
15  ENDFORM.
```

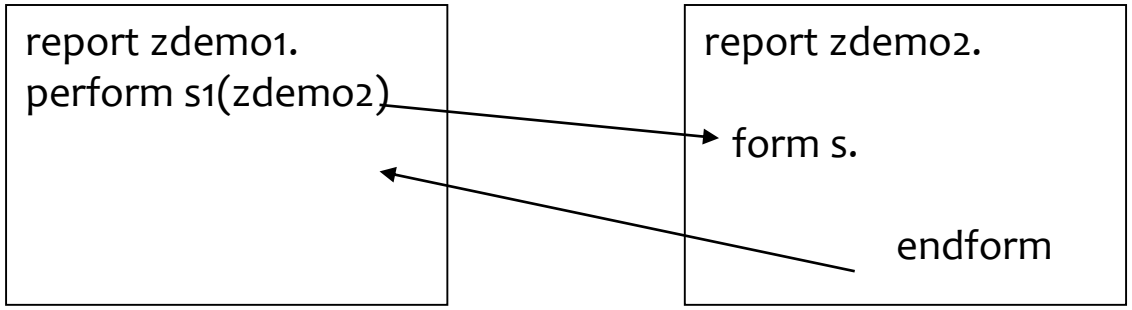
Output after execution

```
Before invoking S2, F1 = A
In Subroutine S2, P1 = X
In EOS,After invoking S2, F1 = A
```

© 2018 Capgemini. All rights reserved.

Defining and Calling External Subroutines

An external subroutine is one that resides in a different program than the perform statement that calls it.
The below figure illustrates an external subroutine.



When a perform calls an external subroutine

- The external program containing the subroutine is loaded
- The entire external program is checked for syntax
- Control transfers to the form in the external program
- The statements within the external subroutine are executed
- The endform transfers control back to the statement following the perform

© 2018 Capgemini. All rights reserved.

Demo1: Call subroutine defined in other program

ABAP Editor: Change Report ZPGM1

Pattern

Report ZPGM1 Active

```
1  REPORT ZPGM1.
2  PERFORM DISP(ZPGM2).
```

ABAP Editor: Change Report ZPGM2

Report ZPGM2 Active

```
1  REPORT ZPGM2.
2  FORM DISP.
3    WRITE 'This is in a different program'.
4  ENDFORM.
```

© 2018 Capgemini. All rights reserved.

Demo2: Call subroutine defined in other program

ABAP Editor: Change Report ZPGM1

Pattern

Report ZPGM1 Active

```
1  REPORT ZPGM1.
2  DATA SNAME(15) TYPE C VALUE 'DISP'.
3  DATA PGMNAME(15) TYPE C VALUE 'ZPGM2'.
4  PERFORM (SNAME) IN PROGRAM (PGMNAME).
```

ABAP Editor: Change Report ZPGM2

Report ZPGM2 Active

```
1  REPORT ZPGM2.
2  FORM DISP.
3    WRITE 'This is subroutine in ZPGM2'.
4  ENDFORM.
```

© 2018 Capgemini. All rights reserved.

Defining and Calling External Subroutines

External subroutines are very similar to internal subroutines:

- Both allow parameters to be passed
- Both allow typed formal parameters
- Both allow parameters to be passed by value, by value and result, and by reference.
- Both allow local variable definitions

© 2018 Capgemini. All rights reserved.

Demo3: Call subroutine defined in other program

ABAP Editor: Change Report ZPGM1

Report ZPGM1 Active

```
1  REPORT ZPGM1 .
2  DATA NUM1 TYPE I VALUE 10.
3  DATA NUM2 TYPE I VALUE 20.
4  DATA RES TYPE I.
5
6  PERFORM ADDNUM(ZPGM2)
7  USING NUM1 NUM2 CHANGING  RES.
8  WRITE :/ 'The Result is:', RES.
```

ABAP Editor: Change Report ZPGM2

Report ZPGM2 Active

```
1  REPORT ZPGM2.
2  FORM ADDNUM USING VALUE(N1) VALUE(N2)
3         CHANGING R.
4    R = N1 + N2.
5  ENDFORM.
```

© 2018 Capgemini. All rights reserved.

Passing Internal Tables as Parameters to Subroutines

Internal tables can be passed as parameters to subroutines.

© 2018 Capgemini. All rights reserved.

Passing an Internal Table without Header Line and automatically creating a header line in Subroutine

If the internal table doesn’t have a header line and you want to pass the body and create a header line in the subroutine, you can also use the syntax

- *form s1 tables it*.

This passes the body by reference, and creates a header line locally in the subroutine.

Changes made to the body of the internal table within the subroutine are immediately reflected in the original.

© 2018 Capgemini. All rights reserved.

Demo: Pass an Internal table without Header Line to a Subroutine

© 2018 Capgemini. All rights reserved.

Passing an Internal Table without Header Line – without automatically creating header line in subroutine

If an internal table doesn’t have a header line and you want to pass the body without automatically creating a header line in the subroutine, you can use the following syntax:

- form s1 using value(it)
- form s1 using it
- form s1 changing it
- form s1 changing value(it)

© 2018 Capgemini. All rights reserved.

Demo: Pass an Internal table without Header Line to a Subroutine

© 2018 Capgemini. All rights reserved.