Capgemini

ABAP Part I

Lesson 05: Common Control statements – Self Study

Lesson Objectives

In this lesson, you will learn about:

- If Statement and Case Statement
- The exit Statement
- The do and the While statement
- The continue and Check statement

IF statement

The if statement in ABAP/4 has relational operators for equality and inequality and special relational operators for string comparisons and bit masks.

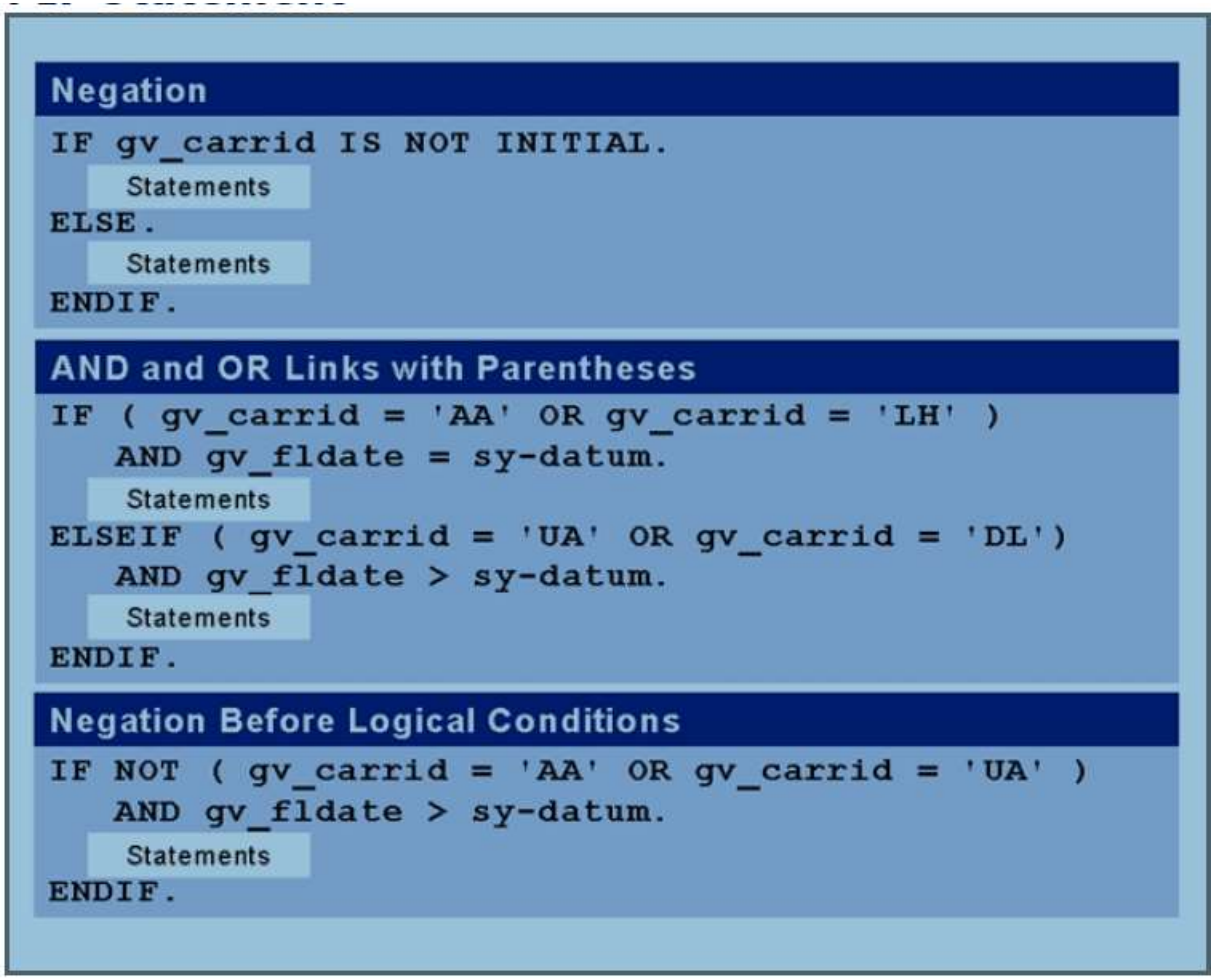
Syntax:

```
        if   [not]  exp [ and [not] exp ] [ or [not] exp ].
   ---
        [elseif  exp.
     ---]
        [else.
   ---]
        endif.
```

where:

- *exp* is a logical expression that evaluates to a true or false condition
- --- represents any number of lines of code.
- Even zeros lines are allowed

IF statement



Negation

```
IF gv_carrid IS NOT INITIAL.
    Statements
ELSE.
    Statements
ENDIF.
```

AND and OR Links with Parentheses

```
IF ( gv_carrid = 'AA' OR gv_carrid = 'LH' )
   AND gv_fldate = sy-datum.
    Statements
ELSEIF ( gv_carrid = 'UA' OR gv_carrid = 'DL')
   AND gv_fldate > sy-datum.
    Statements
ENDIF.
```

Negation Before Logical Conditions

```
IF NOT ( gv_carrid = 'AA' OR gv_carrid = 'UA' )
   AND gv_fldate > sy-datum.
    Statements
ENDIF.
```

Demo

Program on using If Statement

Logical operators for operands of any type

Comparison	Alternate Forms	True When
v1 = v2	eq	v1 equals v2
v1 <> v2	ne, ><	v1 does not equal to v2
v1 > v2	gt	v1 is greater than v2
v1 < v2	lt	vi is less than v2
v1 >= v2	ge, =>	v1 is greater than or equal to
v1 <= v2	le, =<	v1 is less than or equal to
v1 between v2 and v3		v1 lies between v2 and v3 (inclusive)
not v1 between v2 and v3		v1 lies outside of the range v2 and v3 (inclusive)

In the above table v1 and v2 can be variables, or literals, or field strings.

In the case of variables or literals, automatic conversion is performed if the data type or length does not match.

Field strings are treated as type *c* variables.

Case Statement

The *case* statement performs a series of comparisons.

Syntax:

```
case  v1.
        when v2 [ or vn …].
                ---
        when v3 [ or vn …].
                ---
        [when others.
                ---]
        endcase.
```

where:

- *v1* or *v2* can be a variable, literal, constant, or field string
- --- represents any number of line of code.
- Even zero lines are allowed

Case Statement

case is very similar to *if/else*.

The only difference is that on each *if/elseif*, you can specify complex expression.

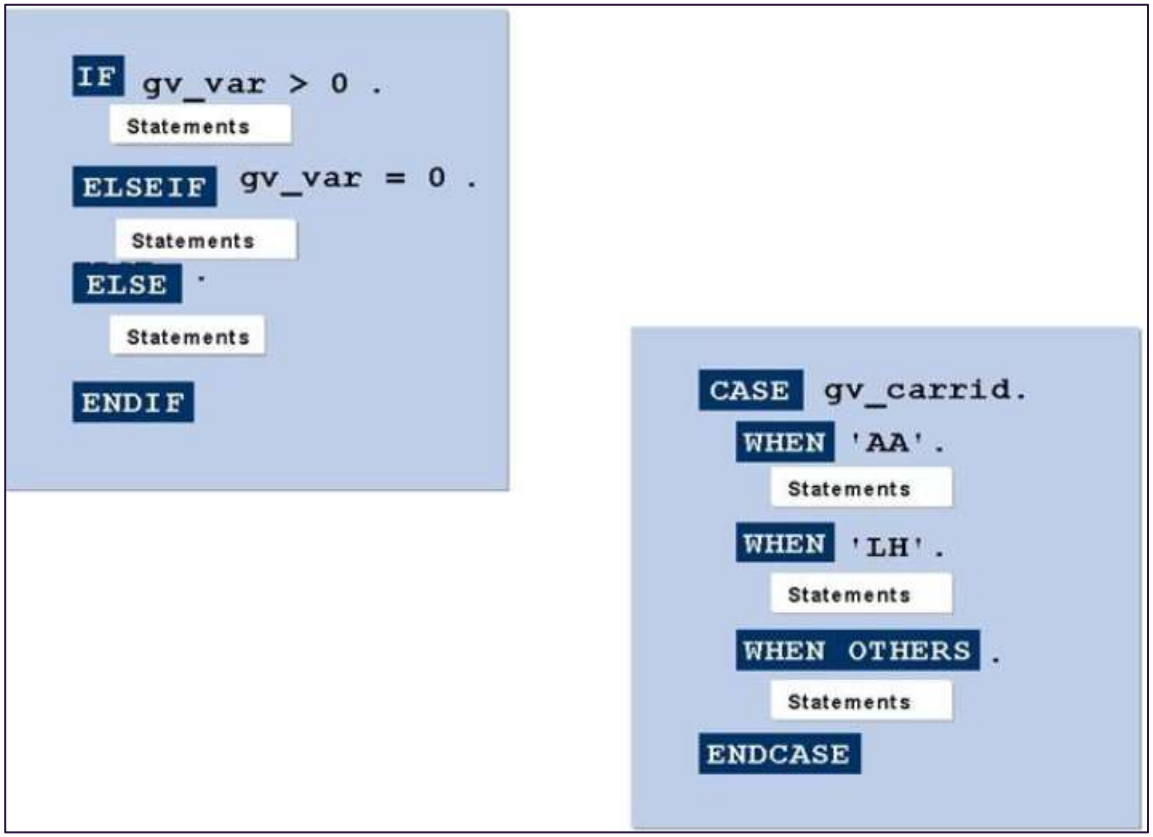
With *case*, you can specify only a single value to be compared, and values are always compared for equality

Demo

Program on using case Statement

IF and Case statement

Conditional Branches



Exit statement

The exit statement prevents further processing from occurring.

Syntax:

exit.

The following example shows a sample program using exit.

```
report zdemo506.
        write: / ‘Hi’.
        exit.
        write: / ‘There’.
```

The above code produces this output:

Hi

Demo

Program on using Exit Statement

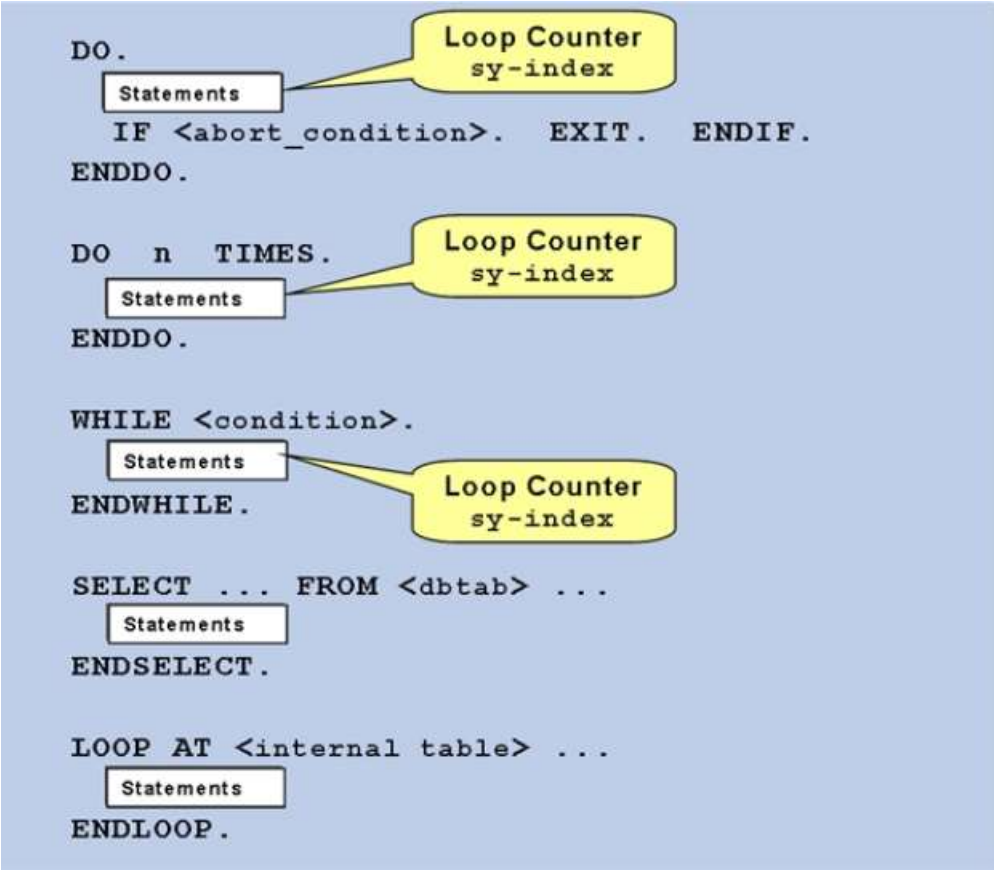
Loops

Unconditional loops using the DO…..ENDDO.

Conditional Loops Using the WHILE ….ENDWHILE.

Loops through Internal Tables using the LOOP….ENDLOOP.

Loops through datasets from database Tables using the SELECT ……ENDSELECT



Do statement

Syntax:

```
do  [ v1 times]
----
[exit.]
----
enddo.
```

where:

- *v1* is a variable, literal, or constant
- ---- represents any number of lines of code

Using the while Statement

The *while* statement is a looping mechanism similar to *do*.

Syntax:

while

[exit.]

----

endwhile

Demo

Program on Loops – Do and While

Loops

Terminating Loops

- Terminating Loop Pass Unconditionally
  - CONTINUE
  - EXIT
- Terminate Loop Pass Conditionally
  - CHECK

Continue Statement

The *continue* statement is coded within a loop.

It acts like a *goto* passing control, immediately to the terminating statement of the loop and beginning a new loop pass.

In effect, it causes the statement below it within the loop to be ignored and a new loop pass to begin.

Continue Statement

Syntax:

It can be used within a *do, while, select*, or *loop*.

[do/while/select/loop]

---

continue.

[enddo/endwhile/endselect/endloop]

where:

--- represents any number of lines of code

Example: Continue statement

The *continue* statement jumps to the end of the loop, ignoring all statements after it for the current loop pass.

Demo

Program on continue statement

Check Statement

The *check* statement is coded within a loop.

It can act very much like *continue*, passing control immediately to the terminating statement of the loop and bypassing the statements between.

Unlike *continue*, it accepts a logical expression.

If the expression is true, it does nothing.

If it is false, it jumps to the end of the loop

Check Statement

Syntax:

It can be used within a *do, while, select*, or *loop*.

[do/while/select/loop]

---

check exp.

[enddo/endwhile/endselect/endloop]

where:

- *exp* is a logical expression
- --- represents any number of lines of code

Check Statement

The *check logic_expr* statement has the following effect:

- Outside a loop, you can terminate a processing block prematurely.
- The block statements after the *check* statement are skipped if the logical condition is not fulfilled (false).
- The system then continues with the first statement in the next processing block
- Within a loop, it has the effect that the next loop is processed.

Check Statement

The check statement is a conditional continue statement.

It jumps to the end of the loop, if the logical expression is false.

If the expression is true, it does nothing.

If it is false, it jumps to the end of the loop

Demo

Program on check statement

Comparing the exit, continue, and check Statements

Statement	Effect
exit	Leaves the current loop
continue	Unconditional jump to the end of the loop
check exp	Jumps to the end of the loop if exp is false

Summary

In this lesson, you have learnt:

- If Statement and Case Statement
- The exit Statement
- The do and the While statement
- The continue and Check statement

Review Question

Question 1: In a case statement a complex expression can be compared

- True/False

Question 2: The _____ statement leaves the current loop.