Условия и циклы языка ABAP

Условия

- **Операторы ветвления** обеспечивают выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.
- В ABAP существуют следующие условные операторы:
  - IF
  - CASE

IF

- Оператор IF позволяет перенаправлять выполнение программы к конкретному объявленному блоку, в зависимости от значения условия. Этот блок состоит из всех команд, которые находятся между объявлением IF и до следующего объявления ELSEIF, ELSE, или ENDIF.

Синтаксис оператора IF

- IF <condition$_1$>.
  <statement block>
  ELSEIF <condition$_2$>.
  <statement block>
  ELSEIF <condition$_3$>.
  <statement block>
  .....
  ELSE.
  <statement block>
  ENDIF.

- Если первое условие истинно, то система выполняет всё до конца первого блока операторов, а затем продолжает обработку после объявления ENDIF.
- Если первое условие ложно, то система обрабатывает следующий ELSEIF таким же образом, что и оператор IF.
- ELSE начинает операторный блок, который выполняется, если ни одно из условий IF или ELSEIF не верно.
- Конец блока условий всегда должен быть завершен оператором ENDIF.

- ABAP позволяет вкладывать блоки IF ... ENDIF на любую глубину. Тем не менее, они должны начинаться и заканчиваться в том же блоке обработки. Другими словами, IF - ENDIF блок не может содержать ключевое слово событий.

Пример работы оператора IF

```
DATA: TEXT1(30) VALUE 'This is the first text',
TEXT2(30) VALUE 'This is the second text',
TEXT3(30) VALUE 'This is the third text',
STRING(5) VALUE 'eco'.

IF TEXT1 CS STRING.
   WRITE / 'Condition 1 is fulfilled'.
ELSEIF TEXT2 CS STRING.
   WRITE / 'Condition 2 is fulfilled'.
ELSEIF TEXT3 CS STRING.
   WRITE / 'Condition 3 is fulfilled'.
ELSE.
   WRITE / 'No condition is fulfilled'.
ENDIF.
```

Вывод выглядит следующим образом:
Условие 2 выполнено.
Здесь второе логическое выражение TEXT2 CS STRING верно, так как строка "eco" содержится в TEXT2

CASE

- Чтобы выполнять различные операторные блоки в зависимости от содержания конкретных полей данных, следует использовать оператор CASE.

Синтаксис оператора CASE

```
CASE <f>.
   WHEN <f1>.
        <statement block>
WHEN <f2>.
        <statement block>
WHEN <f3>.
        <statement block>
WHEN...
  ......
   WHEN OTHERS.
        <statement block>
ENDCASE.
```

- Система выполняет блок операторов после объявления WHEN, если содержимое <F> соответствует <f i >, и продолжает обработку после объявления ENDCASE.
- Блок, объявленный после дополнительного оператора WHEN OTHERS выполняется, если содержимое <F> не равно любому из <Fi>. Оператор CASE должен завершаться оператором ENDCASE.

- Ветвление с использованием CASE является сокращенной формой подобного ветвления с использованием IF:
- ```
  IF <f> = <f1>.
     <statement block>
  ELSEIF <f> = <f2>.
     <statement block>
  ELSEIF <f> = <f3>.
     <statement block>
  ELSEIF <f> =...
    ...
  ELSE.
     <statement block>
  ENDIF.
  ```
  ```

- ABAP позволяет вкладывать блоки CASE... ENDCASE и объединять их с блоками IF... ENDIF . Тем не менее, они должны начинаться и заканчиваться в том же блоке обработки.

Пример

```
DATA: TEXT1 VALUE 'X',
TEXT2 VALUE 'Y',
TEXT3 VALUE 'Z',
STRING VALUE 'A'.
```

```
CASE STRING.
  WHEN TEXT1.
     WRITE: / 'String is', TEXT1.
  WHEN TEXT2.
     WRITE: / 'String is', TEXT2.
  WHEN TEXT3.
     WRITE: / 'String is', TEXT3.
  WHEN OTHERS.
     WRITE: / 'String is not', TEXT1, TEXT2, TEXT3.
ENDCASE.
```

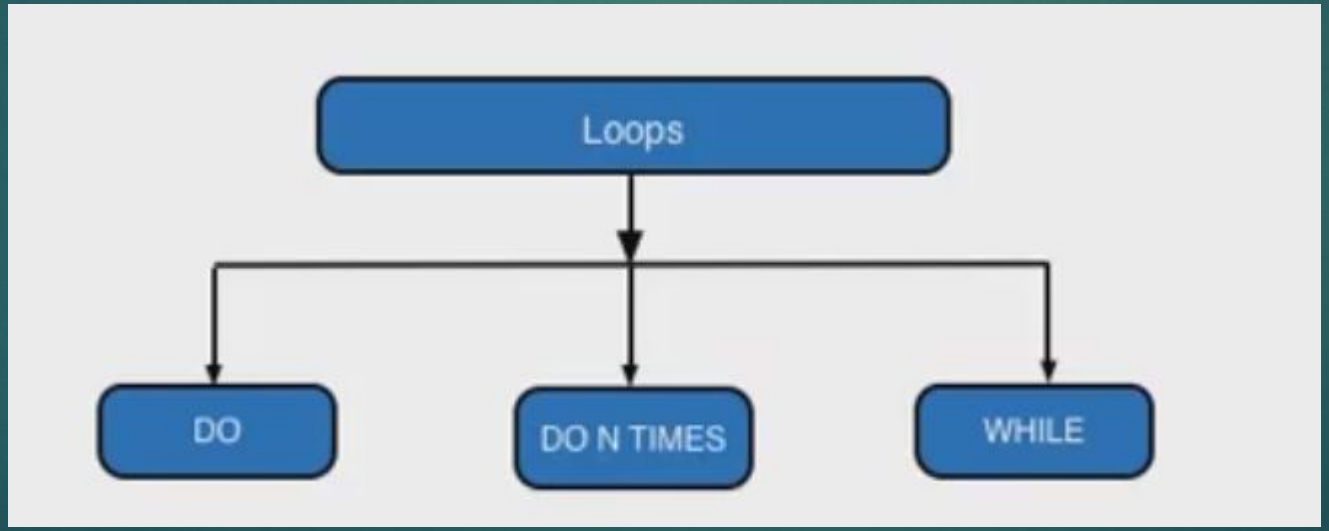
Вывод выглядит следующим образом:
String is not X Y Z

Последний блок операторов после WHEN OTHERS обрабатывается, потому что содержание STRING, "A", не равно "X", "Y", или "Z".

Циклы

В цикле, блок операторов выполняется несколько раз подряд.
Существует три вида циклов в ABAP:

- Безусловные циклы, использующие оператор DO.
- Цикл DO N TIMES.
- Условные циклы, использующие оператор WHILE.



Безусловные циклы

Для обработки блока операторов несколько раз, можно использовать следующую конструкцию:

DO [n TIMES] ...

[statement_block]

ENDDO.

Используют дополнение TIMES чтобы ограничить количество проходов цикла до n.

Если не указать ни каких дополнений, блок оператора будет повторятся до тех пор, пока не достигнет завершающих операторов, таких как EXIT или STOP. Системное поле **sy-index** содержит число проходов цикла, включая проход текущего цикла.

Оператор DO

Синтаксис:

DO.

<statement block>

ENDDO.

Пример:

```
DO.
WRITE sy-index.
 IF sy-index = 3.
 EXIT.
 ENDIF.
ENDDO.
```

На выходе получим:

1 2 3

Здесь цикл проходит обработку 3 раза и затем покидает ее после оператора EXIT.

Пример с двумя вложенными циклами с дополнением TIMES

```
DO 2 TIMES.
WRITE sy-index.
 SKIP.
 DO 3 TIMES.
 WRITE sy-index.
 ENDDO.
 SKIP.
ENDDO.
```

На выходе получим:

```
1
1 2 3
2
1 2 3
```

Внешний цикл обрабатывается дважды, внутренний цикл обрабатывается три раза. Системное поле **sy-index** содержит номер прохода для каждого цикла в отдельности.

Условные циклы

Чтобы повторить блок операторов, пока определенное условие истинно, используют следующую конструкцию:

WHILE log_exp

[statemaent_block]

ENDWHILE.

log_exp может быть любым логическим выражением. Блок операторов между WHILE и ENDWHILE повторяется пока условие верно или до тех пор, пока не достигнет завершающих операторов таких как EXIT или STOP.

Можно использовать вложенные циклы **WHILE** на любую глубину, и объединять их с другой формой цикла.

Пример условного цикла

```
REPORT demo_flow_control_while.
DATA: length  TYPE i VALUE 0,
 strl  TYPE i VALUE 0,
 string(30) TYPE c VALUE 'Test String'.
strl = strlen( string ).
WHILE string NE space.
 WRITE string(1).
 length = sy-index.
 SHIFT string.
ENDWHILE.
WRITE: / 'STRLEN: ', strl.
WRITE: / 'Length of string:', length.
```

Вывод выглядит следующим образом:

```
T e s t  S t r i n g
STRLEN: 11
Length of String: 11
```

Здесь цикл **WHILE** используется для определения длины строки символов. Это достигается путем сдвига строки на одну позицию влево при каждом входе в цикл и обрабатывается до тех пор, пока он содержит только пробелы.

Завершающиеся циклы

ABAP содержит завершающие операторы, которые позволяют завершить цикл досрочно. Существует две категории завершающих операторов – те, которые применяются только к циклу, и те, которые распространяются на весь блок обработки, в котором происходит цикл. Операторы STOP и REJECT относятся к последней группе.

Завершающие операторы, которые применяются только к циклу, в котором они произведены это CONTINUE, CHECK и EXIT. Можно использовать только оператор CONTINUE. CHECK и EXIT, с другой стороны, являются контекстно-зависимыми. В цикле они применяются только для выполнения самого цикла. За пределами цикла, они завершают весь процесс обработки блока, в котором они возникают (подпрограммы, диалог, модуль, событие блока, и так далее).

CONTINUE, CHECK и EXIT могут быть использованы во всех видах цикла в ABAP (DO, DO N TIMES, WHILE).

Безусловное завершение прохождения цикла

Чтобы сразу и безусловно завершить прохождение простого цикла, используется оператор CONTINUE в блоке цикла.

После заявления, система игнорирует все оставшиеся инструкции в нынешнем блоке, и начинается прохождение следующего цикла.

Пример:

```
DO 4 TIMES.
 IF sy-index = 2.
 CONTINUE.
 ENDIF.
 WRITE sy-index.
ENDDO.
```

На выходе получим:

1 3 4

Второе прохождение цикла завершается без оператора WRITE.

Условное завершение прохождения цикла

Чтобы завершить простой цикл условно, используют оператор CHECK condition в блоке операторов цикла.

Если условие не истинно, все оставшиеся операторы в нынешнем блоке после оператора CHECK игнорируются, и начинается следующее прохождение цикла, условием(condition) может быть любое логическое выражение.

```
DO 4 TIMES.
 CHECK sy-index BETWEEN 2 and 3.
 WRITE sy-index.
ENDDO.
```

На выходе получим:

2 3

Первое и четвертое прохождение цикла прекращаются без оператора WRITE, потому что **sy-index** находится между 2 и 3.

Выход из цикла

Чтобы завершить весь цикл сразу и безусловно, используется оператор EXIT в блоке операторов цикла.

После этого оператора, цикл завершается, и обработка возобновляется после закрытия оператора цикла структуры(ENDDO, ENDWHILE, ENDLOOP, ENDSELECT). Во вложенных циклах, только текущий цикл завершается.

```
DO 4 TIMES.
 IF sy-index = 3.
 EXIT.
 ENDIF.
 WRITE sy-index.
ENDDO.
```

На выходе получим:

1 2

В третьем прохождении цикла, цикл завершает работу, прежде написать оператора WRITE.