Синтаксис ОСА.

ОСА содержит интерпретатором несложного языка, с помощью которого можно проводить логический контроль, автоматические перекодировки признаков по условиям,
строить новые признаки в массиве по заранее подготовленной схеме.

Синтаксические задания могут готовиться как с помощью редактора ОСА, так и с помощью любого другого текстового редактора. Задания могут содержаться блоки заданий разного типа. Рассмотрим их.

Блок построения новых признаков.
Это блок находится между тегами <sign> и </sign> и предназначен для построения новых признаков, на основе уже имеющихся.

Формат записи блока следующий:

Название признака
Текст вопроса
Тип шкалы (n,o,m или j. Если шкала j – необходимо через пробел указать количество альтернатив)
Список альтернатив
***
Условия (в формате ОСА for Windows)
/
Иначе (указывается обязательно для всех шкал).

Пример:
<sign>
Мой признак
Мой вопрос
J 2
1 Первая альтернатива
2 Первая альтернатива
***
#1#=1 -> 1
#1#=5 -> 2
/
$

Мой признак
Мой вопрос
m
***
#1#=1 -> 1.4
#1#=5 -> 2.8
/
6
</sign>

Следует заметить, что новые признаки можно строить на основе ранее созданных в том же самом задании.

Пример:
<sign>
Мой признак
Мой вопрос
m
***
#[Number]#=1 -> 1.4
#[Number]#=5 -> 2.8
/
6
</sign>

В этом задании идет обращение к признаку с именем Number.
В паспорте имя признака необходимо указать в комментариях к признаку в формате:
@qname=Название_Признака

Пример:
Номер анкеты
Номер анкеты
m
***
@qname=Number
/
Регистр букв при этом не имеет значения. Названия можно давать как на английском, так и на русском языках.

Составляя задание, Вы можете вместо номера признака использовать его имя. Это позволяет использовать стандартные для Вас перекодировки в любом массиве, составив их один раз. Имя признака записывается в квадратных скобках. Конечно, не следует использовать квадратные скобки в именах признаков.

Блок логического контроля.
Это блок находится между тегами <logc> и </logc> и предназначен для логического контроля данные и перекодировок имеющихся признаков.

При запуске этого блока автоматически находится ключевой признак (признак помеченный директивой @key, если таких несколько берется тот, который имеет минимальный номер).
В дальнейшем, при выводе информации о найденных ошибках и т.п. помимо физического номера анкеты в массиве выводится также и значения ключевого признака. Обычно в качестве ключевого признака используют номер анкеты.

Каждая строка блока представляет собой одно условие в следующем формате:

X -> Y
где X условие в формате ОСА (возможно использование имен), Y любая тестовая информация, которая будет выводиться при выполнении условия.

Пример:
n(6) -> НеОтвет в признаке 6
E([Num],6) and v(3)=1 -> в признаке [Num] значение 6 и при этом в значение 3 в признаке 1.

Кроме того, в условиях поддерживаются одноуровневые циклы.
Формат их записи следующий:

For *a* to *b*; *c*: X -> Y

где *a* - стартовое значение цикла, *b* – значение, при превышении которого цикл заканчивает свой работу, *c* - шаг с которым меняется *a* при каждом выполнении цикла. Для того чтобы обратиться в условии к текущему значению *a*, необходимо использовать команду *!index*. При этом *a,b,c* должны быть целыми числами.

Рассмотрим пример.

For 1 to 3; 1: n(!index) -> НеОтвет в признаке !index

Эта запись эквивалентна следующему набору условий

n(1) -> НеОтвет в признаке 1
n(2) -> НеОтвет в признаке 2
n(3) -> НеОтвет в признаке 3

Еще один пример.

For 2 to 6; 2: n(index) -> НеОтвет в признаке index

Эта запись эквивалентна следующему набору условий

n(2) -> НеОтвет в признаке 2
n(4) -> НеОтвет в признаке 4
n(6) -> НеОтвет в признаке 6

Возможно определение и использование переменных в программах логического контроля.
Переменные задаются командой SET
SET x=y
где x – имя переменной, а y ее значение. Внимание! Имя переменной обязательно должно начинаться символом !
В качестве значения переменной может быть указано либо число, либо формула.
Область видимости переменной весь файл задания.

Пример
SET !a=1
V(!a)=5 -> Ошибка! В признаке !a введено 5.

Эквивалентно записи
V(1)=5 -> Ошибка! В признаке 1 введено 5.

Пример
SET !a=v(5)+6
n(!a)=5 -> Ошибка! В признаке !a введено 5.

Эквивалентно записи

НЕТ
Если в переменной равной значению переменной номер 5 плюс 6 записано значение НеОтвет будет сообщение о ошибке.

Использование переменных удобно при написании длинных повторяющихся условий где многократно фигурируют одни и те же признаки. В этом случае для изменения номера или имени признака достаточно только поменять значение переменной, а не искать и исправлять все вхождения в тексте программы.

Строки, начинающиеся с точки с запятой, считаются комментариями и игнорируются

Также в блоке логического контроля могут находиться команды по автоматическому исправлению логических ошибок. В этом случае строка-условие должно выглядеть так:
X ->> Y
Как мы видим, немного поменялся вид стрелки разделяющей левую и правую часть.

В правой части (обозначаемой нами как Y) возможны следующие команды:
PUT *e,f*
PUTKEY *e,f*
ADD *e,f*
DEL *e,f*
COPY *e,f*
CCAJ *e,f*
TXCRECODE *e,f*
Команда PUT переписывает имеющийся код в признаке с номером (именем) *e* на новый код *f.*

Команда PUTKEY переписывает имеющийся код в признаке с номером (именем) *e* на новый код *f.* При этом она вносит соответствующие изменения в файл текстовых замечаний, исправляя записи, для которых признак с номером (именем) *e* указан, как ключевой.

Команда ADD добавляет к имеющимся кодам в признаке с номером (именем) *e* ответа новый код *f* (корректно только для признаков с совместимыми альтернативами).

Команда DEL удаляет из отмеченных в признаке с номером (именем) *e* кодов код *f* .

Команда COPY копирует значение признака номер *e* в признак с номером *f*. Шкалы признаков должны совпадать.

Команда CCAJ в признаке с номером (именем) *e* ответа оставляет ответов не более чем *f* (корректно только для признаков с совместимыми альтернативами), лишние ответы удаляются, выбираясь случайным, равновероятным образом.

Команда TXCRECODE вносит изменения в файл текстовых замечаний (txc-файл). Комментарии, относящиеся к признаку с номером *e,* переписываются как комментарии, относящиеся к признаку с номером *f.*

Пример:
#1#=5 ->> Put [mysign],2
В этом случае в анкетах, у которых в качестве ответа на признак 1 введена 5, в признаке с именем mysign будет записана 2.

Также возможна команда перекодировки:
EXCH *e,x -> z*
где *e* - номер (имя) признака, *x* – код 1, *z* – код 2. Эта команда заменяет в признаке с номером (именем) *e* код 1 на код 2.

Пример:
#1#=2 ->> EXCH [mysign],2 -> 4
ВНИМАНИЕ! Данная команда не может быть использована в метрических признаках и признаках-каталогах.

Кроме того, в правой части можно также использовать команду SET для изменения значения переменных.

Пример:
SET !x=5
#1#=2 ->> SET !x=!x+10

Можно указать к выполнению несколько команд, разделив их точкой с запятой.

Пример:
#1#=2 ->> Put [mysign],2; Put 2,5

Команда HIDE позволяет не выводить в листинг сообщение о выполненном действии для одной строки. Это бывает необходимо, для проведения, например, некоторых дополнительных расчетов для логического контроля.
Пример:
For 1 to 5;1: #!index# = 1 ->> hide; set !x=!x+1
#!x#<3 -> мало ответов!

В блоке логического контроля допустимы некоторые специальные команды.
NumberCheck – вызывает функцию проверки контроля номеров, содержащихся в ключевом признаке. Контролируется отсутствие НеОтветов в ключевом признаке, последовательность нумерации, повторы номеров и отсутствующие номера.
GotoCheck – Эта команда вызывает функцию автоматической проверки условий переходов, описанных в паспорте с помощью директивы @goto. Команду лучше ставить сразу после тега <logc>, до начала описаний условий контроля и перекодировки.
GotoCheckRec – Эта команда вызывает функцию автоматической проверки условий переходов, описанных в паспорте с помощью директивы @goto. При нахождении ошибки не только сообщается о ней, но и производится автоматическая перекодировка признаков, на значения, описанные в директиве. Команду лучше ставить сразу после тега <logc>, до начала описаний условий контроля и перекодировки.
Аналогично с тегом @goto есть команды и для проверки условий прописаных с помощью тега @jump - JumpCheck и JumpCheckRec.
RangeCheck - эта команда вызывает функцию автоматической проверки условий, описанных в паспорте с помощью директивы @R. Команду лучше ставить сразу после тега <logc>, до начала описаний условий контроля и перекодировки.
NoWorkList – эта команда запрещает вывод в листинг сообщений о выполненных перекодировках и найденных ошибках, что, при проведении масштабных перекодировок, дает прирост скорости выполнения файла-задания.

Сообщения о командах, которые вносят изменения в массив, выводятся в листинг синим цветом.

Блок управления массивом.
Это блок находится между тегами <arrwrk> и </arrwrk>.
В этом блоке могут быть следующие команды.
OpenArray=X – открывает для работы массив с именем X. Если указано имя без полного пути к массиву, то массив ищется в том же каталоге, где находится текущий открытый массив. Ранее открытый массив при этом закрывается. Указывать расширение файла-массива не обязательно.
Пример:
OpenArray=c:\arrays\1.frm
OpenArray=c:\arrays\1

CloseArray – закрывает текущий открытый массив.
Строки, начинающиеся с точки с запятой, считаются комментариями и игнорируются

DelSigns=X –удаляет из открытого на текущий момент масиива признаки с указанными номерами (именами). Несколько признаков следует указывать через пробел. Несколько подряд идущих признаков можно указать через тире.
Пример:
DelSigns=1 2 5-10
DelSigns=[start]-[end]
При перемещении признаков НЕ вносятся изменения в файл текстовых замечания (txc-файл). Если необходимо такие изменения вносить используйте команду DelSignsTXC.

CopyArray=X; Y – создает копию текущего загруженного массива с именем массива X и именем паспортаY. Копия создается в той же папке, где находится открытый массив. Указывать расширения для имен файлов необязательно. Если файлы с такими именами уже существуют они, будут переписаны.
Пример:
CopyArray=newarr;newpass
CopyArray=newarr.frm;newpass.dcl

CopyFile=X -> Y – копирует файл с именем X в файл с именем Y. Если файл с именем Y существует он переписывается. Если не указан путь к файлу, берется путь к текущему загруженному массиву.

DelFile=X – удаляет файл с именем X, если таковой имеется. Если не указан путь к файлу, берется путь к текущему загруженному массиву.

MoveSign=X -> Y – переставляет признаки в загруженном массиве. X – список признаков (если несколько признаков, следует указывать через пробел; если несколько признаков идущих подряд, используйте тире). Y – признак после которого следует поместить перемещаемые признаки. Если в качестве параметра Y указать 0 признаки будут перемещены в самое начало массива. При перемещении признаков НЕ вносятся изменения в файл текстовых замечания (txc-файл). Если необходимо такие изменения вносить используйте команду MoveSignTXC.

Svm=X Y Z K – данная команда проводит реструктуризацию массива, создавая массив дневников (подробнее см. документацию к OCA New Line Technology раздел “Функция Реструктурирование данных” -> “Создание дневниковых массивов”). Параметры: X – номер

признака начинающего первый блок, Y- номер признака, заканчивающего первый блок, Z - количество блоков, K – номер ключевого признака (того, по которому будет идентифицироваться в дальнейшем дневник). После окончания работы открытым становится НОВЫЙ массив, что дает Вам возможность скопировать его под новым именем.