Session 3: More Logic in Data Entry – Rosters, Subscripts and Loops

At the end of this lesson participants will be able to:

- Use subscripts, totocc(), count(), seek() and curocc() to work with rosters
- Use loops (*do while*) to implement edit checks on repeating records/items (rosters)
- Understand the order of PROCs and where to place edit checks
- Understand and use "special values" (notappl, missing and default) in logic

Groups and Subscripts

Let's add a check to the mothers IDNO number field in section C to ensure that the line number entered corresponds to a woman 12 or over in the section B roster. To do this we need to link the line number entered in section C back to the corresponding row of the roster in section B. This is done through subscripts. For any item that is repeated, adding a number in parentheses after it gives you the value of a particular occurrence of the variable. For example AGE(3) gives you the age of the third row of the roster.

```
PROC CHILD_MOTHER

// Ensure that line number of mother is line number of woman over 12
// in section B roster.
if not CHILD_MOTHER in 87:88 then

    if SEX(CHILD_MOTHER) = 1 then
        errmsg("Sex of mother %s is male",
               strip(NAME(CHILD_MOTHER)))
               select("Correct mother IDNO",
                      CHILD_MOTHER,
                      "Correct sex of " + strip(NAME(CHILD_MOTHER)),
                      SEX(CHILD_MOTHER));
    endif;

    if AGE(CHILD_MOTHER) < 12 then
        errmsg("Age of mother %s is %d but must be at least 12",
               strip(NAME(CHILD_MOTHER)), AGE(CHILD_MOTHER))
               select("Correct mother IDNO", CHILD_MOTHER,
                      "Correct age of " + strip(NAME(CHILD_MOTHER)),
                      AGE(CHILD_MOTHER));
    endif;
endif;
```

Blanks and special values

What happens if we refer to an occurrence of a row in the section B roster that doesn't exist? Try entering an IDNO for the mother greater than the number of rows in the section B roster. Our tests for age and sex are not triggered. What are the values of NAME, SEX and AGE when they are empty? Let's print them out using *errmsg* and see.

```
errmsg("NAME = %s, AGE=%d, SEX=%d", strip(NAME(CHILD_MOTHER)),
AGE(CHILD_MOTHER), SEX(CHILD_MOTHER));
```

The alphanumeric variable NAME is just empty but the numeric values AGE and SEX are NOTAPPL. What does this mean? Generally numeric fields that are blank (skipped or not yet entered) have a special value called *notappl* that can be used in comparisons in logic. For example to test if the SEX is blank we can use the following comparison:

```
    if SEX(CHILD_MOTHER) = notappl then
        // Sex is blank, must be an empty row in section B roster
        errmsg("%d is not a valid line in section B", CHILD_MOTHER);
        reenter;
    endif;
```

There are other special values that are used in CSPro:

- Missing: can be used as an alias for no response/refused codes (9, 99, 999…). You must create an entry for it in the value set.
- Default: results from an error reading from a data file or from a calculation error (like trying to calculate notappl + 2).

Getting the size of a roster

In our error message it would be nice to tell the interviewer what the maximum valid line number is. We can do that using the function *totocc()* which gives you the total number of occurrences of a group.

```
    if SEX(CHILD_MOTHER) = notappl then
        // Sex is blank, must be an empty row in section B roster
        errmsg("%d is not a valid line in section B. Must be between 1 and
%d.",
            CHILD_MOTHER, totocc(ADULTS000));
        reenter;
    endif;
```

Group exercise

Display an error message if the father (in section B) is not at least 12 years OLDER than his child in section C. You should do this check in the father IDNO field (C10). You will need to convert between months and years to be able to compare the age of the father and the age of the child.

Getting the current row number

When we are in the proc of a variable in a group we don't need to specify the occurrence number using a subscript. For example, in the proc for AGE we can use the variables AGE and RELATIONSHIP without subscripts. CSPro knows which row of the roster we are in and automatically uses the AGE and RELATIONSHIP from the current row. But if we want to know the current row number, for example to

automatically fill in the line number in the adults roster, we can use the function *curocc()* which returns the current occurrence of a group.

```
PROC LINE_NUMBER

preproc
// Automatically fill in line number based on current row number of roster
LINE_NUMBER = curocc();
```

We can now make the line number field protected.

More About Procs

It is currently possible to enter multiple heads of household or no head of household at all. How can we add a consistency check to ensure that there is exactly one head of household? In which proc would such a check go?

Not only do variables have procs but there are procs for forms, rosters, levels and even the application itself. Everything you see in the forms tree can have both a preproc and a postproc. Understanding the order in which procs are executed is important in understanding CSPro logic. The general rule is that parent items have their preproc called first, then the procs of the child items are called and finally the postproc of the parent is called.

Group Exercise: Proc Order

Form teams of 3-5 people. The instructor adds errmsg to the postproc and preproc of the following: SURVEY_FF (application), SURVEY_QUEST (level), ADULTS_FORM (form), ADULTS000 (roster),NAME (variable), ROOMS (variable).

Each team receives slips of paper with the names of each preproc and postproc. BEFORE running the application the teams have to put the slips in the order that the errmsgs will be shown when the application is run. Teams have three minutes to complete the exercise. Then the application is run and teams see if they had the correct results.

Variables

Returning to our check on the number of heads of household, which proc should the check go in? We can put it in the postproc of the roster since that will be run after all the rows have been entered.

How do we determine the number of heads of household? In each row of the roster we can check the relationship to see if it is head of household. To do this we need to introduce logic variables which are like dictionary variables but they are only for use in calculations in logic and don't get shown on forms or saved to the data file. To create a variable we declare it as follows:

```
numeric aNumber;
string anAlphanumeric;
alpha(20) anAlphaNumericWithAFixedLength;
```

Coding style

We recommend using mixed case (a.k.a. CamelCase) for logic variables in order to distinguish them from dictionary variables.

How do we declare a variable to keep track of the number of heads of household?

```
numeric numberOfHeads = 0;
```

Note that we can declare and give it a value at the same time.

Now we can increment the variable for each head of household we find:

```
PROC ADULTS000

// After adults roster is complete, check to make sure that there is
// exactly one head of household.

numeric numberOfHeads = 0;

if RELATIONSHIP(1) = 1 then
    numberOfHeads = numberOfHeads + 1;
endif;

if RELATIONSHIP(2) = 1 then
    numberOfHeads = numberOfHeads + 1;
endif;

if RELATIONSHIP(3) = 1 then
    numberOfHeads = numberOfHeads + 1;
endif;

if numberOfHeads <> 1 then
    errmsg("Number of heads of household must be exactly one");
    reenter RELATIONSHIP(1);
endif;
```

Note that since there is more than one row in the roster we need to use a subscript to tell CSEntry which row of the roster to look in. RELATIONSHIP(3) refers to the relationship of the 3rd row of the roster.

Loops

The above works but only if we know the size of the household in advance. In order to handle any size household we need a loop. We can use a do loop which lets you repeat something until a certain

condition is true. How many times do we loop? We use the function *totocc()* that gives us the total number of occurrences of a repeating record or item.

```
PROC ADULTS000

// After adults roster is complete, check to make sure that there is
// exactly one head of household.

numeric numberOfHeads = 0;
numeric i;
do i = 1 while i <= totocc(ADULTS000)
    if RELATIONSHIP(i) = 1 then
        numberOfHeads = numberOfHeads + 1;
    endif;
enddo;

if numberOfHeads <> 1 then
    errmsg("Number of heads of household must be exactly one");
    reenter RELATIONSHIP(1);
endif;
```

Activity: Acting out a loop

Show a household with 4 members on the whiteboard/flipchart and show the code above on the screen. Pick 3 volunteers. One volunteer plays the role of the variable numberOfHeads, a second plays the role of the variable i and a third is the loop director. Give them each a sign so everyone knows who they are playing. The two people playing variables should show their current values by holding up the correct number of fingers. The director is responsible for walking through the code line by line and updating the variable values until the loop ends explaining what they do step by step.

Now that we all understand loops let's see how we can simplify our code by using the function *count()* instead of a loop.

```
PROC ADULTS000

// After adults roster is complete, check to make sure that there is
// exactly one head of household.

numeric numberOfHeads = count(ADULTS where RELATIONSHIP = 1);

if numberOfHeads <> 1 then
    errmsg("Number of heads of household must be exactly one");
    reenter RELATIONSHIP(1);
endif;
```

Let's take a second example. Show an error message if the head of household is less than 12 years older than any of his/her children.

```
// Make sure that the age of the head of household is at least 12
// years greater than the age of his/her children

// First find the index of the head.
numeric indexHead;
numeric i;
do i = 1 while i <= totocc(ADULTS000)
    if RELATIONSHIP(i) = 1 then
        indexHead = i;
        break; // quit loop early
    endif;
enddo;

// Compare age of head to age of each child
do i = 1 while i <= totocc(ADULTS000)
    if RELATIONSHIP(i) = 3 and AGE(indexHead) - AGE(i) < 12 then
        errmsg("Child %s is %d years old but head %s is %d. Parent must be at
least 12 years older than child.",
                strip(NAME(i)),
                AGE(i),
                strip(NAME(indexHead)),
                AGE(indexHead))
                select("Correct age of " + strip(NAME(i)), AGE(i),
                       "Correct age of " + strip(NAME(indexHead)),
AGE(indexHead),
                       "Correct relationship of " + strip(NAME(i)),
RELATIONSHIP(i),
                       "Correct relationship of " + strip(NAME(indexHead)),
RELATIONSHIP(indexHead));
    endif;
enddo;
```

We can simplify this code a little by using the function *seek()*.

```
// First find the index of the head.
numeric indexHead = seek(RELATIONSHIP = 1);
```

Exercises

1. Display an error message in the IDNO field of section D if the line number given does not correspond to the line number of a woman over 12 in the section B roster.
2. Automatically fill in the line number in the children roster and make the field protected.
3. Show an error message if the head of household has at least one spouse in the household and the marital of status of the head of household is not married.
4. Show an error message if the head of household and his/her spouse are of the same gender. Make sure that your logic works with polygamous households as well as households with just one spouse.