Session 4: More Logic in Data Entry, Rosters, Subscripts and Loops

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

- Understand the order of PROCs and where to place edit checks
- 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 and use "special values" (notappl, missing and default) in logic

Groups and Subscripts

Let's add a check to the **line number of mother** to ensure that the line number entered corresponds to a woman 12 or over in the household. To do this we need to link the line number entered in B13 back to the corresponding row of the roster in section B. This is done through subscripts to get the age and sex of the mother based on the line number.

```
PROC MOTHER_LINE_NUMBER

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

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

    if AGE(MOTHER_LINE_NUMBER) <> 999 and AGE(MOTHER_LINE_NUMBER) < 12 then
        errmsg("Age of mother %s is %d but must be at least 12",
               strip(NAME(MOTHER_LINE_NUMBER)),
               AGE(MOTHER_LINE_NUMBER))
               select("Correct mother", MOTHER_LINE_NUMBER,
                      "Correct age of " + strip(NAME(MOTHER_LINE_NUMBER)),
                      AGE(MOTHER_LINE_NUMBER));
    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 a line number 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(MOTHER_LINE_NUMBER)),
        AGE(MOTHER_LINE_NUMBER), SEX(MOTHER_LINE_NUMBER));
```

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(MOTHERS_LINE_NUMBER) = notappl then
        // Sex is blank, must be an empty row in section B roster
        errmsg("%d is not a valid line in section B", MOTHERS_LINE_NUMBER);
        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 or n/0 (divide by zero)) or moving a field into a field that is too small to hold the value. For example, if AGE is a two digit field, AGE = 118 will result in ***Default***.

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 MOTHER_LINE_NUMBER > totocc(DEMOGRAPHICS_ROSTER) then
        // This is beyond the end of the roster
        errmsg("%d is not a valid line in section B. Must be between 1 and
%d.",
            MOTHER_LINE_NUMBER, totocc(DEMOGRAPHICS_ROSTER));
        reenter;
    endif;
```

Group exercise

Display an error message if the father line number (B15) is the line number of an individual that is not male or is not at least 12 years old. You should do this check in the father line number field.

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

1. Parent items have their **preproc** called first,
2. then the **procs** of the child items are called
3. and finally, the **postproc** of the parent is called.

Group Exercise: Proc Order

Form teams of three to five people. The instructor adds errmsg to the postproc and preproc of the following: POPSTAN2020_FF (application), POPSTAN2020_QUEST (level), DEMOGRAPHICS_FORM (form), NAMES_ROSTER (roster), NAME (variable), NUMBER_OF_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 have the correct results.

Counting Heads of Household

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 a logic 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 DEMOGRAPHICS_ROSTER

// After person 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;
```

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 DEMOGRAPHICS_ROSTER

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

numeric numberOfHeads = 0;
do numeric i = 1 while i <= totocc(DEMOGRAPHICS_ROSTER)
    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 four members on the whiteboard/flipchart and show the code above on the screen. Pick three 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 DEMOGRAPHICS_ROSTER

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

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

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

Going back to our earlier check of mother and father line numbers, it turns out that our check does not work in all cases. If we enter the parent before the child it works fine but if we enter the child first, then the check fails because we don't yet know the age and sex of the parent. We can fix this by moving it to the postproc of the demographics roster too and again using a loop.

```
// Ensure that line number of mother is line number of woman over 12
// in section B roster.
do numeric i = 1 while i <= totocc(DEMOGRAPHICS_ROSTER)
    if not MOTHER_LINE_NUMBER(i) in 87:88 then

        if SEX(MOTHER_LINE_NUMBER(i)) = 1 then
            errmsg("%s is mother of %s but has sex male",
                   strip(NAME(MOTHER_LINE_NUMBER(i))),
                   strip(NAME(i)))
                   select("Correct mother of " + strip(NAME(i)),
                          MOTHER_LINE_NUMBER(i),
                          "Correct sex of " +
                                 strip(NAME(MOTHER_LINE_NUMBER(i))),
                          SEX(MOTHER_LINE_NUMBER));
        endif;

        if AGE(MOTHER_LINE_NUMBER(i)) <> 999 and
           AGE(MOTHER_LINE_NUMBER(i)) < 12 then
            errmsg("%s is mother of %s but is less than 12",
                   strip(NAME(MOTHER_LINE_NUMBER(i))),
                   strip(NAME(i)))
                   select("Correct mother of " + strip(NAME(i)),
                          MOTHER_LINE_NUMBER(i),
                          "Correct age of " +
                              strip(NAME(MOTHER_LINE_NUMBER(i))),
                          AGE(MOTHER_LINE_NUMBER(i)));
        endif;

    endif;
enddo;
```

The father line number check is similar.

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

```
// Make sure that the age of the head of household is at least twelve
// 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(DEMOGRAPHICS_ROSTER)
    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(DEMOGRAPHICS_ROSTER)
    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. 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.

2. 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.

3. Ensure that the age difference between the head of household and the grandchild should not be less than 24 years. If age of head is X years and the age of the child is Y years, then X-Y ≥ 24.