Session 6: Navigation & System Control

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

- Create user defined functions in CSPro
- Extend the interface of CSEntry with userbar buttons
- Use the commands *advance* and *move* to navigate through the questionnaire
- Use the function *visualvalue* to get the value of variables that are "off path"
- Use the command *showarray* to display tables to the interviewer

User defined functions

Often you find you have identical blocks of logic in multiple procs in your application. This can cause problems if you later change the code in one place to fix a bug and forget to change it the other. In such situations it is better to put the logic in a user defined function which you can then call from all the procs where it is used. User defined functions are defined in the PROC GLOBAL. You call them the same way you call built in CSPro functions.

User defined functions can take arguments and return values just like built in functions. Let's define a function that we can use in all the places where we check if a household member is a woman of childbearing age. This function will be passed the index (row number) of the household member in the section B roster and it will return one if the person a woman over 12 years old and zero otherwise. This way if we later decide that we should be using 13 or 14 as a minimum age we only have to make the change in one place.

```
// Determine if member of household from Adults
// roster is a woman of childbearing age.
// Pass in the index (occurence number) of the
// of the household member.
function isChildbearingWoman(index)
    if SEX(index) = 2 and AGE(index) >= 12 then
        isChildbearingWoman = 1;
    else
        isChildbearingWoman = 0;
    endif;
end;
```

Now we can use this function when building the value sets for mother of child (C8) and idno of the woman in section D.

```
PROC CHILD_MOTHER
onfocus
// Create the value set for child mother from all eligible
// women in section B
numeric i;
```

```
numeric nextEntry = 1;
do i = 1 while i <= totocc(ADULTS000)

    if isChildbearingWoman(i) = 1 then
        labels(nextEntry) = NAME(i);
        codes(nextEntry) = i;
        nextEntry = nextEntry + 1;
    endif;
enddo;
labels(nextEntry) = "non-resident";
codes(nextEntry) = 87;
nextEntry = nextEntry + 1;
labels(nextEntry) = "dead";
codes(nextEntry) = 88;
nextEntry = nextEntry + 1;
codes(nextEntry) = notappl;
setvalueset(CHILD_MOTHER, codes, labels);
```

```
PROC FERTILITY_WOMAN_ID
onfocus
// Create the value set for fertility from all eligible
// women in section B
numeric i;
numeric nextEntry = 1;
do i = 1 while i <= totocc(ADULTS000)

    if isChildbearingWoman(i) = 1 then
        // Don't allow the same person twice
        if seek(FERTILITY_WOMAN_ID = i) = 0 then
            labels(nextEntry) = NAME(i);
            codes(nextEntry) = i;
            nextEntry = nextEntry + 1;
        endif;
    endif;
enddo;
codes(nextEntry) = notappl;
setvalueset(FERTILITY_WOMAN_ID, codes, labels);
```

We can also create a function to avoid repeating the same dynamic value set code in each of the occupation subitems. If you examine the procs for levels 2, 3 and 4 of the occupation code they only differ in what is passed to *getlabel()* and the name of the item passed to *setvalueset()*. We can leave the setvalueset() call outside our function and pass in the portion of the argument to *getlabel()* that varies from field to field.

```
// Fill in codes, labels array for occupation value set.
// Pass the first 1,2 or 3 digits as the baseCode and
// the resulting value set will have all the sub-categories
// of the baseCode. For example if the base code is 12
// the value set will have all the values that start
// with 12 (121, 122, 123...).
function createOccupationCodeValueSet(baseCode)
    // Create value set by appending digits 1-9 to the base code.
```

```
    numeric i;
    numeric nextEntry = 1;
    do i = 0 while i <= 9
        string label = getlabel(OCCUPATION, baseCode * 10 + i);
        // A blank label means that there is no code for that number. Only
        // add an entry if there is a label for the code.
        if label <> "" then
            codes(nextEntry) = i;
            labels(nextEntry) = label;
            nextEntry = nextEntry + 1;
        endif;
    enddo;
    codes(nextEntry) = notappl;
end
```

Using this function in the three occupation fields results in much less code:

```
PROC OCCUPATION_LEVEL_2
onfocus
// Create value set based on occupation level 1 chosen in previous question.
createOccupationCodeValueSet(OCCUPATION_LEVEL_1);
setvalueset(OCCUPATION_LEVEL_2, codes, labels);

PROC OCCUPATION_LEVEL_3
onfocus
// Create value set based on level 1 and 2 categories selected in previous
question
createOccupationCodeValueSet(OCCUPATION_LEVEL_1 * 10 + OCCUPATION_LEVEL_2);
setvalueset(OCCUPATION_LEVEL_3, codes, labels);

PROC OCCUPATION_LEVEL_4
onfocus
// Create value set based on level 1,2 and 3 categories selected in previous
question
createOccupationCodeValueSet(OCCUPATION_LEVEL_1 * 100 +
                OCCUPATION_LEVEL_2 * 10 +
                OCCUPATION_LEVEL_3);
setvalueset(OCCUPATION_LEVEL_4, codes, labels);
```

The Userbar

From logic we can add buttons to the CSEntry user interface that call user-defined functions. Here is how to add a userbar button that will create an errmsg dialog that says “hello”. First we define the function hello in the PROC GLOBAL:

```
function hello()
    errmsg("Hello");
end;
```

Then in the preproc of the application we add it to the userbar:

```
PROC SURVEY_FF
preproc
userbar(clear);
userbar(add button, "Hello", hello);
userbar(show);
```

When adding a button in the preproc of the application it is important to call *clear* first, otherwise we can end up with two or three copies of the same button if we start the application multiple times. We added the button in the preproc of the application but you can add or remove buttons in any proc so you can, for example, only a show button within a certain field or a certain roster.

Let's try a more interesting example. Let's add a "Go To…" button that will let the user navigate directly to a particular section of the questionnaire.

```
// Userbar function for navigating
// directly to different parts of survey.
function goto()
    numeric section = accept("Go to?",
                             "Identification",
                             "Members 5 and over",
                             "Members under 5");
    if section = 1 then
        skip to HOUSEHOLD_IDENTIFICATION;
    elseif section = 2 then
        skip to ADULTS_FORM;
    elseif section = 3 then
        skip to CHILDREN_FORM;
    endif;
end;
```

```
PROC SURVEY_FF
preproc
userbar(clear);
userbar(add button, "Go To...", goto);
userbar(show);
```

Advance and Move

The above will let the interviewer jump from one section to another but using skip means that if we use our Go To… to jump over a section, that section will end up skipped and won't be saved in the data file. Instead of using *skip* we can use *advance* which moves forward in the questionnaire without marking fields as skipped. Using advance also runs all the preprocs and postprocs of the fields that are passed through to ensure that no consistency or out of ranges checks are missed.

```
// Userbar function for navigating
// directly to different parts of survey.
function goto()
    numeric section = accept("Go to?",
```

```
                                "Identification",
                                "Members 5 and over",
                                "Members under 5");
    if section = 1 then
        advance to HOUSEHOLD_IDENTIFICATION;
    elseif section = 2 then
        advance to ADULTS_FORM;
    elseif section = 3 then
        advance to CHILDREN_FORM;
    endif;
end;
```

This stops from skipping over data when navigating however it still has a limitation. We can only navigate forward in the questionnaire. To go backwards we need to use reenter but in our *goto()* function we don't know if the user wants to move forward or backward. Fortunately CSPro provides the command *move* which will use either *skip* or *reenter* as appropriate. By default, when going forward, move does a skip but you can add *advance* after the field name to make it do an advance instead of a skip.

```
// Userbar function for navigating
// directly to different parts of survey.
function goto()
    numeric section = accept("Go to?",
                                "Identification",
                                "Members 5 and over",
                                "Members under 5");
    if section = 1 then
        move to HOUSEHOLD_IDENTIFICATION advance;
    elseif section = 2 then
        move to ADULTS_FORM advance;
    elseif section = 3 then
        move to CHILDREN_FORM advance;
    endif;
end;
```

Let's extend our goto function to let the interviewer navigate directly to an individual in the section B roster. If they choose section B then instead of going to the first person in the roster, we will show them a list of all household members in the roster and the let them choose which one to goto. For this we can use the function *showarray*(). This function takes an array of values and displays them in a grid in a dialog box and returns the row number that the user picks.

Unlike *setvalueset*() that takes two arrays, *showarray()* takes a two-dimensional array. You can think of a two-dimensional array as a grid of variables or as matrix. You declare a two-dimensional array in the PROC GLOBAL the same way you declare a one-dimensional array except that you specify the size in both dimensions: number of rows and number of columns.

```
array string adultsArray(30, 3);
```

In our case we want up to 30 rows, one for each person, and we will use 3 columns so that we can display the name, sex and relationship for each person.

When assigning a value to a two-dimensional array you specify both the row and column you want to put the value in:

```
// Set value in row 4, column 2
adultsArray(4, 2) = "This is the 4th row, 2nd column";
```

In our examples we will loop through the adults in the section B roster and add the name, sex and relationship for each one to our array. Note that for sex and relationship we want the value set labels so we use *getlabel()*. Since this will be more than a few lines of code let's put this into a function by itself and then call it from our *goto()* function.

```
// Show list of entries in adults roster in a dialog
// and let interviewer pick one.
// Returns the row number of the person that was picked
// or zero if the dialog was cancelled.
function pickFromAdultsRoster()
    numeric i;
    do i = 1 while i <= totocc(ADULTS000);
        adultsArray(i, 1) = strip(NAME(i));
        adultsArray(i, 2) = getlabel(SEX, SEX(i));
        adultsArray(i, 3) = getlabel(RELATIONSHIP, RELATIONSHIP(i));
    enddo;
    adultsArray(i, 1) = "";
    numeric picked = showarray(adultsArray, title("Name", "Sex",
"Relationship"));
    pickFromAdultsRoster = picked;
end;

// Userbar function for navigating
// directly to different parts of survey.
function goto()
    numeric section = accept("Go to?",
                             "Identification",
                             "Members 5 and over",
                             "Members under 5");
    if section = 1 then
        move to HOUSEHOLD_IDENTIFICATION advance;
    elseif section = 2 then
        numeric index = pickFromAdultsRoster();
        if index > 0 then
            move to NAME(index) advance;
        endif;
    elseif section = 3 then
        move to CHILDREN_FORM advance;
    endif;
end;
```

Path and Visualvalue

Our *goto()* function works when we are in section C or D but when we are in section A the sex and relationship are blank. What is going on? In system controlled mode, CSEntry keeps track of which variables are on and off the "path". Variables that you have entered are considered "on path" but those that have been skipped, even if there was a value in them before they were skipped, are considered "off path". As we have seen before, variables that are "off path" are considered blank (*notappl*) in logic. It turns out that variables that are ahead of the current field are also considered "off path" until you pass through them. The idea is that these fields have not yet been validated by running their preproc and postproc with the current values of all preceding fields and therefore cannot be considered final. The effect of this is that the values of all fields ahead of the current field in the questionnaire are *notappl* in logic. Interestingly, as we can see from our current code, this only applies to numeric items. Our code works just fine for the NAME field.

You can see which fields are "on path" by looking at the background color of the field:

- Green: on path
- Dark Grey: skipped
- White: not yet been filled in
- Light grey: protected

Note that the field coloring scheme is different in operator controlled mode.

Fortunately, we can get the value of fields that are "off path" using the function *visualvalue()*. It returns whatever value is currently visible in the field whether or not it has been skipped or is ahead of the current field. Using this for relationship and sex in our function we get:

```
function pickFromAdultsRoster()
    numeric i;
    do i = 1 while i <= totocc(ADULTS000);
        adultsArray(i, 1) = strip(NAME(i));
        adultsArray(i, 2) = getlabel(SEX,
                                     visualvalue(SEX(i)));
        adultsArray(i, 3) = getlabel(RELATIONSHIP,
                                     visualvalue(RELATIONSHIP(i)));
    enddo;
    adultsArray(i, 1) = ""; // mark end of array
    numeric picked = showarray(adultsArray,
                               title("Name", "Sex", "Relationship"));
    pickFromAdultsRoster = picked;
end;
```

All that is left now is to use the appropriate relationship for the sex of the household member:

```
if visualvalue(SEX(i)) = 1 then
    adultsArray(i, 3) = getlabel(RELATIONSHIP_MALE,
                                 visualvalue(RELATIONSHIP(i)));
else
```

```
    adultsArray(i, 3) = getlabel(RELATIONSHIP_FEMALE,
                                 visualvalue(RELATIONSHIP(i)));
endif;
```

Exercises

1. Extend the *goto()* function to include the remaining sections of the questionnaire (D through H).
2. Extend the goto() function so that when the interviewer chooses section C (children) the application displays a table with the name of the child, sex, age in months, name of mother and name of father. Make sure to handle the case where the mother or father is dead or non-resident. When you the interviewer chooses one of the children, go the first field in the chosen row of the children roster.
3. Add a button to the userbar called "household summary" that when clicked uses the errmsg function to display a message that shows the name of the head of household, the number of household members by sex. For example:

   "Head of Household: John Brown, Total Members: 5, Women: 2, Men: 3".

   Make sure to include both adults and children in the totals.

   Bonus if you can get this to work when you click the button from section A. Hint: count and seek will not work with visualvalue so you will need to use a loop to find the number of males of and females from sections A and C.
4. In the postproc of the adults roster we currently check that the head of household is at least 12 years older than all of the other household members. Add the same check for the spouse. In other words check that the spouse of the head of household is at least 12 years older than all the children of the head of household. Create a single function that is used for this consistency check for both the head and for the spouse(s). It should probably be a soft check to handle the case of children from second marriages.
5. Add a button to the userbar that is only visible while in the fertility section called "goto woman". Clicking on this button should send you to the NAME field in the section B roster for the woman whose fertility is currently being entered. To make it visible only in the fertility section you will need use the *add* option of the *userbar()* command in the onfocus of the roster and the *remove* option of the *userbar()* command in the killfocus. See the help for more information on these options.