Session 7: Functions, Navigation and Visualvalue

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 in 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 eligible for labor. This function will be passed the index (row number) of the household member in the demographic roster and it will return one if the person is over 15 years old and zero otherwise. This way if we later decide that we should be using 13 or 14 as a minimum age, or we want to use other variables in our eligibility criteria we only have to make the change in one place.

```
// Determine if member of household from demographics
// roster is eligible for labor section.
// Pass in the index (occurrence number) of the
// of the household member in the demographics roster.
function isEligibleForLabor(personNumber)
    if DEMAG1(personNumber) >= 15 then
        isEligibleForLabor = 1;
    else
        isEligibleForLabor = 0;
    endif;
end;
```

Now we can use this function in ask if clauses and when building the value set for **LBPPNO**.

```
// Module universe - persons of working age
if not isEligibleForLabor(LBPPNO) then
    skip to PXYEND;
else
    PXYSTART = 1;
endif;
```

```
PROC LBPPNO

onfocus

// Build value set from all eligible members of household roster
numeric nextValueSetEntry = 1;
do numeric nextPerson = 1 while nextPerson <= totocc(DEM_ROSTER)
    // Check if eligible for labor section
    numeric existingOccInLaborRoster = seek(LBPPNO = PPNO(nextPerson));
    if isEligibleForLabor(nextPerson) and existingOccInLaborRoster = 0 or
existingOccInLaborRoster >= curocc() then
        // Add to value set
        codes(nextValueSetEntry) = PPNO(nextPerson);
        labels(nextValueSetEntry) = NAME(nextPerson);
        nextValueSetEntry = nextValueSetEntry + 1;
    endif;
enddo;
codes(nextValueSetEntry) = notappl;
setvalueset(LBPPNO, codes, labels);
```

Group Exercise

Add a user defined function checkboxHasSelection() that takes as a parameter a string value from a checkbox field and returns 1 if at least one item in it is selected and zero if no items are selected. Use this function in the postprocs of AGFCHECK and ACDYSYM to replace the expression in the if condition for checking if the checkbox has a selection (*length(strip(XXXX) = 0*). Note that to have a user defined function accept a string parameter instead of a numeric parameter your function declaration will need to include the parameter type:

```
    function checkboxHasSelection(string checkbox)
```

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 POPSTANLFS_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 questionnaire.
function goto()
    numeric section = accept("Go to?",
                             "Identification",
                             "Demographics",
                             "Labor");
    if section = 1 then
        skip to GH_FORM;
    elseif section = 2 then
        skip to DEM_FORM;
    elseif section = 3 then
        skip to LABOR_FORM;
    endif;
end;
```

```
PROC POPSTANLFS_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 range checks are missed.

```
// Userbar function for navigating
// directly to different parts of questionnaire.
function goto()
    numeric section = accept("Go to?",
                             "Identification",
                             "Demographics",
                             "Labor");
    if section = 1 then
        advance to GH_FORM;
    elseif section = 2 then
        advance to DEM_FORM;
    elseif section = 3 then
```

```
        advance to LABOR_FORM;
    endif;
end;
```

This stops the function 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 questionnaire.
function goto()
    numeric section = accept("Go to?",
                             "Identification",
                             "Demographics",
                             "Labor");
    if section = 1 then
        move to GH_FORM advance;
    elseif section = 2 then
        move to DEM_FORM advance;
    elseif section = 3 then
        move to LABOR_FORM advance;
    endif;
end;
```

Let's extend our goto function to let the interviewer navigate directly to an individual in the demographics roster. If they choose "demographics" 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 go to. 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 householdMembersArray(20, 3);
```

In our case we want up to 20 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
householdMembersArray(4, 2) = "This is the 4th row, 2nd column";
```

In our examples, we will loop through the members in the household 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 household 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 pickFromHouseholdRoster()
    numeric i;
    do i = 1 while i <= totocc(DEM_ROSTER)
        householdMembersArray(i, 1) = strip(NAME(i));
        householdMembersArray(i, 2) = getlabel(DEMSEX_VS1, DEMSEX(i));
        householdMembersArray(i, 3) = getlabel(DEMREL_VS1, DEMREL(i));
    enddo;
    householdMembersArray(i, 1) = ""; // Mark end
    numeric picked = showarray(householdMembersArray, title("Name", "Sex",
"Relationship"));
    pickFromHouseholdRoster = picked;
end;

// Userbar function for navigating
// directly to different parts of questionnaire.
function goto()
    numeric section = accept("Go to?",
                             "Identification",
                             "Demographics",
                             "Labor");
    if section = 1 then
        move to GH_FORM advance;
    elseif section = 2 then
        numeric index = pickFromHouseholdRoster ();
        if index > 0 then
            move to NAME(index) advance;
        endif;
    elseif section = 3 then
        move to LABOR_FORM advance;
    endif;
end;
```

Path and Visualvalue

Our *goto()* function works when we are in the labor section but when we are in the identification form, 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 pickFromHouseholdRoster()
    numeric i;
    do i = 1 while i <= totocc(DEMOGRAPHICS_ROSTER);
        householdMembersArray(i, 1) = strip(NAME(i));
        householdMembersArray(i, 2) = getlabel(DEMSEX_VS1,
visualvalue(DEMSEX(i)));
        householdMembersArray(i, 3) = getlabel(DEMREL_VS1,
                                              visualvalue(DEMREL(i)));
    enddo;
    householdMembersArray(i, 1) = ""; // Mark end
    numeric picked = showarray(householdMembersArray,
                               title("Name", "Sex", "Relationship"));
    pickFromHouseholdRoster = picked;
end;
```

Group Exercises

1. Add the age to the household member list displayed by the pickFromHouseholdRosterFunction.

2. Change the pickFromHouseholdRoster function to show the correct label for the gender of the household member in the household list that is displayed. Instead of showing "Bart Male Son/Daughter" and "Mary Female Father/Mother" it should show "Bart Male Son" and "Mary Female Mother" respectively.

Setting field values only once

Let's prefill the interview start date. We can make the field protected and set the value in the preproc just like we did with the PPNO field. We can use the function *sysdate*() which returns the current date as a number formatted according to the format specification passed in.

```
PROC GHVFDT
preproc
// Prefill interview date with current system date.
GHVFDT = sysdate("YYYYMMDD");
```

This works the first time we visit the field but what happens we come back to the question a day or two later? We only want to record this value the first time the interviewer enters the field and once it is set we don't want it to change. We can do this by comparing the value of GHVFDT to blank (notappl) in the preproc and only setting the value to sysdate if it is blank.

```
PROC GHVFDT
preproc
// Prefill interview date with current system date.
if GHVFDT = notappl then
    GHVFDT = sysdate("YYYYMMDD");
endif;
```

But this doesn't seem to work. Why? Is GHVFDT on path when we are in the preproc of GHVFDT? It is not. We have to get to the postproc for the variable to be on path. However, we can use *visualvalue*() to get the value of the field in the preproc:

```
PROC GHVFDT
preproc
// Prefill interview date with current system date.
if visualvalue(GHVFDT) = notappl then
    GHVFDT = sysdate("YYYYMMDD");
endif;
```

Exercises

1. Extend the *goto()* function to allow the user to choose which individual to jump to in the labor section. Instead of moving to the start of the labor form, use showarray to display the list of household members in the labor section and move to LBPPNO for the individual selected.
2. Extend the *goto*() function so that when the interviewer chooses to go to the labor section and then to an individual in the labor section, instead of moving to LBPPNO, use accept to show a list of modules (Proxy, Employment, Agriculture…) and jump to the start of the module that is chosen. Bonus: only list modules that have not been skipped over.
3. Add logic to set the occurrence labels for the LABOR_FORM. Handle both the case where a new case is being entered and where you are modifying an existing case as we did with the demographic roster occurrence labels. Hint: you will need to use visualvalue. Make sure to test in the case where you partially save the case while in LBPPNO before filling in the value for that field.
4. Add a warning if, when going to back to the LBPPNO field after entering labor for multiple individuals, the interviewer chooses to change the LBPPNO to the same value as another household member whose labor information is in a later occurrence of the labor form. For

example, if the user enters labor for John, Mary, Jack and Jill and then goes back to the LBPPNO field for John and chooses the line number of Jill they should receive a warning saying that they already entered labor information for Jill. This should be a soft check in case they want to swap the labor information for John and Jill.

5. 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, and the number of household members by sex. For example:

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

   Bonus if you can get this to work when you click the button from the first form (geographic and household identifiers). Hint: count will not work with visualvalue so you will need to use a loop to find the number of males of and females.