

Andy Peredery Con't

Pets: Simba the Cat.

Partner/Family: My lovely wife, Helen.

Sports/Hobbies: Volleyball, pole vaulting, billiards and coin collecting.

Ideal weekend would be: Drive up to Algonquin Park in the fall with Helen and friends, hike a five-hour trail to get the appetite going and finish off with a hearty Hungarian meal!

Favourite Foods:

- Korean Dol-Sot Bi-Bim-Bap (rice, vegetables, marinated beef and one egg (sunny side up) in a steaming hot “stone bowl” ...YUMMY!!).
- Domino’s vegetarian pizza with lots and lots of garlic ☺!
- Any Ukrainian food! (As you can probably tell, I love to eat!)
- ICE CREAM!!!!!!

If I could be anything at all (besides a SAS consultant), I would be ... a Home Depot customer service representative (lumber department) — I love working with wood — or a general contractor for a home builder.

When I’m not involved in SAS consulting projects, I like to ... build stuff out of wood! My last project was to build a custom CD/DVD/VCR tape stand. All of the stands that Wal-Mart, Best Buy, etc., normally carry are too small, or not sturdy enough to hold all of the movies and music I have. For this project, I stained the plywood with a cherry finish and even had special smoked glass doors made for the front!

Something I’ve learned out in the field that would benefit other SAS users is ...the pointer control in the DATA step and PUT/INPUT statements. Pointers keep track of the current position in a file and give you precise control over READ/WRITE operations.

The Basics:

Pointer Control	Example	What it does...
@	“@20” moves to column 20 in the current line.	EXACT column pointer that moves to a specific column in the current line or record.
+	“+(-3)” moves three characters to the LEFT in the current line.	RELATIVE column pointer that moves within the current line or record.
#	“#16” moves to line 16 (column 1) in the file.	EXACT row pointer that moves to the specific row, column 1. Can only move FORWARD (not backwards) from the current row.
/	“/” moves to the next line (column 1).	ADVANCES row pointer to the next line, column 1.

@ (trailing at-sign)	PUT var1 @;	Leaves the column pointer on the CURRENT line but is released and set to the next row (column 1) at the bottom of the observation loop.
@@ (trailing double at-sign)	INPUT var2 @@;	Leaves the column pointer on the CURRENT line and holds it for the next observation to use.

I have found pointer controls invaluable when working with the SAS/IntrNet solution because they give me fine control over the HTML output from a **DATA _null_** step, especially when building URLs. These techniques can also be used for regular Base SAS output reporting.

Normally when SAS outputs variables in a PUT statement, it pads them with a trailing space, even if we've trimmed the variable. (String constants like "Hello world!", on the other hand, are not padded.)

Real-Life Example

The following code snippet was used in one of the SAS/IntrNet projects we built for a SAS customer. I've modified the example slightly to simplify our code walk-through. You'll need SAS/IntrNet installed if you want this example to work completely, but it can be run in Base SAS, and it will create a **_webout.dat** file on your computer locally. The pointer concepts discussed are applicable to both environments.

In our example, the SAS customer wanted a quick and easy-to-use Web search for its client base: a quick click on a letter of the alphabet that would then take them to a results page.

```

01 /* search.sas
02  *SAS/Intrnet – HTML output with pointer control
03  */
04
05 /* display letters A-Z quick links across the top of a web page */
06 data _null_;
07 file _webout;
08   input c $ @@; /* read in the data lines below */
09   put '<a target="results_frame" href="/scripts/broker.exe' @;
10   put '?_service=default' @;
11   put '&_program=mylib1.results.sas' @;
12   put '&search=' c $ +(-1) '>' c $1. '</a>&nbsp;';
13
14 datalines;
15 A B C D E F G H I J K L M
16 N O P Q R S T U V W X Y Z
17 ;
18 run;
```

This is what the generated web page would look like...



Code Walk-Through

Lines 1-5 are comments that we can skip.

Lines 6-7 set up the DATA step to output the HTML code to a Web browser, via a special file called **_webout**. The SAS/IntrNet solution knows to stream this output to the browser.

Line 8 is where our first example of pointer controls begins. We start by reading in some data. Here, we use the “trailing double at-sign” pointer control to stay on the same line of the data (lines 15-16) for each loop of the DATA step. SAS is smart enough to know that when we reach the end of the line (EOL) then the pointer control should be reset to column 1 of the next line.

Lines 9-12 write out a single line of HTML. For lines 9-12, we want the strings to be appended together to build the URL properly. Note that line 12 does not have a trailing at-sign because we want the next HTML line to be written out on a new line.

Line 12 has another pointer control trick that’s very handy when dealing with variables. As mentioned above, SAS outputs variables with a trailing space. In line 12 we use the **+(-1)** pointer control to “edit out” that space when displaying the **c** variable. We could have also used the **\$1.** to format and display the variable exactly, without trailing spaces, as we have done for the second variable output on line 12.

The advantage of the **+(-1)** is that you can support variable-length data without having to use the **\$\$\$.** fixed-length formats.

For example, if line 15 was replaced with the following five groupings:
ABC DEF GHIJ KL M

And line 12 was changed to:

```
put '&search=' %trim(c) +(-1) '>' %trim(c) +(-1) '</a>&nbsp;';
```

Then we’ve effectively made our code handle variables with dynamic length strings. This is where pointer controls really show their power!