December, 2004

Advisor Answers

A LIST Replacement

VFP 9/8/7/6

Q: I've always used the LIST command when I just need to get a quick list of data to send someone. But it no longer works as well as it used to. It leaves way too much space between fields. Is there are way to fix this?

–Advisor DevCon attendee

A: The LIST command (and its nearly identical twin, DISPLAY) has been in the FoxPro language from the beginning. It has many forms, but the simplest is just LIST, which displays the data from the current table. LIST is an Xbase command, which means it can handle scope (NEXT, REST, etc.) as well as a FOR clause. You can specify the fields to list, decide whether record numbers are included, and a number of other options. You can also send LIST output to a file, making it easy to drop into an email, or to the printer, for a very basic report. While LIST isn't something you'd use in an application, it can be very handy for quick-and-dirty results.

Somewhere along the way (perhaps because of the introduction of proportional fonts), the output from LIST became much less attractive. Instead of putting fields into a space that matches the field width, the output from LIST includes lots of extra white space between fields. Unfortunately, there's no way to tell LIST how much space to use for each field.

However, there is a way to get decent-looking columnar output without using a report. I considered several possibilities before settling on the code below. The first thing I tried was sending LIST output to a file, then reading the file in with FileToString() and using either Reduce() from FoxTools (which I discussed in the August issue) or STRTRAN() to get rid of the extra spaces. However, because the amount of extra space depends on the length of the actual data in the field, doing this right would have required a lot of tricky adjustments.

My next approach used the COPY TO command. I figured one of the many text formats it can produce should either give the desired results, or be easily transformed by string manipulation into the

desired results. As with LIST, though, it turned out that variations in the length of the data values in a given column made the task more complex than I expected.

I finally realized that one way or another, I was going to have to address that variation, so I might as well tackle it directly, while the data was still in a VFP table. The plan of attack in the code below is to figure out the number of characters needed to display the desired data for each field, and then to output the data using that many characters.

To make the code useful, it seemed that a function was called for. I call it CleanList. It has one required parameter, the name of the file where the listing is to be stored. The function also accepts four optional parameters. The first is the alias to list—if it's omitted, the current work area is used. The second optional parameter is the list of fields to include. If this parameter is omitted, all fields except General and Blob fields are included in the result. The next parameter is a filter expression—only those records meeting the specified condition are included in the listing. Because it's easy enough to turn a scope expression (like NEXT 5) into a filter condition, I chose not to have a scope parameter. The final parameter is a logical value that determines whether the output is echoed to the screen—it corresponds to the NOCONSOLE clause of the LIST command. Pass .T. to prevent echoing of output.

The function uses a query to put the specified fields of the specified records into a cursor. It then loops through the fields of the cursor and determines the maximum space needed to display for each field, based on the actual data. That is, if you have a 50-character field, but the largest entry has only 12 characters, that field will use 12 columns in the result.

For some data types, like character and memo, figuring out the maximum space needed is straightforward. For others, it's a little more complex. The trickiest is for numeric and currency fields, where you presumably want the decimal points to line up in the result. In that case, the function calculates the maximum width needed for the integer portion and the maximum width needed for the decimal portion separately. Here's the code that computes maximum field width.

```
nFieldCount = AFIELDS(aFldList, "ReportData")
DIMENSION aFldWidth[nFieldCount,2]
FOR nField = 1 TO nFieldCount
   DO CASE
   CASE INLIST(aFldList[nField, 2], "C","V","M")
      * String field, find max width
```

```
      SELECT MAX(LEN(EVALUATE(aFldList[nField,1]))) ;
         FROM ReportData INTO ARRAY aMax
      aFldWidth[nField,1] = aMax[1]

   CASE INLIST(aFldList[nField,2], "G","W")
      * Eliminate fields that can't be displayed
      aFldWidth[nField,1] = 0

   CASE INLIST(aFldList[nField,2], "B", "I")
      * For packed numeric types,
      * figure out maximum length needed
      SELECT MAX(LEN(TRANSFORM( ;
                 EVALUATE(aFldList[nField, 1])))) ;
         FROM ReportData INTO ARRAY aMax
      aFldWidth[nField,1] = aMax[1]

   CASE INLIST(aFldList[nField,2], "N", "Y")
      * Separate decimals and integer to get max size
      SELECT MAX(LEN(TRANSFORM(INT(;
                 EVALUATE(aFldList[nField, 1])))))), ;
             MAX(LEN(SUBSTR(TRANSFORM(;
                 EVALUATE(aFldList[nField,1])), ;
                 AT(SET("POINT"),TRANSFORM(;
                 EVALUATE(aFldList[nField,1])))+1))) ;
         FROM ReportData INTO ARRAY aMax
      aFldWidth[nField,1] = aMax[1]
      aFldWidth[nField,2] = aMax[2]

   CASE aFldList[nField,2] = "T"
      * Make datetime wide enough
      aFldWidth[nField,1] = 20

   CASE aFldList[nField,2] = "L"
      aFldWidth[nField,1] = 3

   OTHERWISE
      aFldWidth[nfield,1] = aFldList[nField,3]
   ENDCASE

ENDFOR
```

Once we know how much space to allocate for each field, generating the output is simple. The function uses an old capability, SET ALTERNATE, to capture the generated output and put it in a file. The function SCANs through the records in the cursor, looping through the fields of each record. For most types, it uses PADR() to pad the data to the maximum size for the field. As in computing field widths, numeric and currency fields require extra attention. The function uses PADL() to right-justify the integer portion, and then PADR() to left-justify the decimal.

Here's the code that generates the output:

```
SET ALTERNATE TO (cOutFile)
SET ALTERNATE ON
SELECT ReportData
nOldMemoWidth=SET("Memowidth")
SET MEMOWIDTH TO 8192
  && maximum allowed, try to avoid wrapping
SCAN
   * Loop through fields,
   * adding one space after each for spacing
   FOR nField = 1 TO nFieldCount
      DO CASE
      CASE aFldWidth[nField,1] = 0
         * Do nothing
      CASE aFldList[nField,2] = "L"
         * Can't use PADR() with logicals
         ?? EVALUATE(aFldList[nField,1]), " "

      CASE aFldList[nField,2] = "N"
         * Special handling for number
         ?? PADL(INT(EVALUATE(aFldList[nField,1])), ;
                aFldWidth[nField,1])
         ?? SET("POINT")
         ?? PADR(SUBSTR(TRANSFORM(;
                 EVALUATE(aFldList[nField,1])), ;
                 AT(SET("POINT"),TRANSFORM(;
                 EVALUATE(aFldList[nField,1])))+1), ;
                 aFldWidth[nField,2] + 1)

      CASE aFldList[nField,2] = "Y"
         * Special handling for currency
         ?? PADL(INT(MTON(EVALUATE(aFldList[nField,1]))), ;
                aFldWidth[nField,1])
         ?? SET("POINT")
         ?? PADR(SUBSTR(TRANSFORM( ;
                 EVALUATE(aFldList[nField,1])), ;
                 AT(SET("POINT"),TRANSFORM( ;
                 EVALUATE(aFldList[nField,1])))+1), ;
                 aFldWidth[nField,2] + 1)

      OTHERWISE
         ?? PADR(EVALUATE(aFldList[nField,1]), ;
                aFldWidth[nField,1] + 1)
      ENDCASE
   ENDFOR
   ?
ENDSCAN

SET ALTERNATE off
SET ALTERNATE TO
```

The complete function is included as CleanList.PRG on this month's Professional Resource CD and in the downloads for this issue.

–Tamar