Many programmers learned programming using BASIC, Fortran, or similar languages prior to exposure to SAS. The adjustment to SAS can be confusing at first. Two problems are an insufficient understanding of the flow of control which the SAS Supervisor imposes on the DATA step, and confusion between external files, SAS data sets, and arrays.

SAS can easily be made to act more like a traditional language, with the use of the DATA NULL statement and the STOP statement. These statements prevent SAS from automatically creating SAS data sets and from executing the implied OUTPUT;RETURN; sequence at the end of each data step.

SAS Data Sets. The SAS Supervisor creates a SAS data set by default in every data step that has an INPUT, SET, UPDATE, or MERGE statement. Programmers wishing to transfer skills learned in other languages, which are common to other languages, may be helpful to stop SAS from its automatic creation of SAS data sets. Use of the special reserved data set name NULL in the statement "DATA NULL;" accomplishes this.

Flow of Control. In Fortran, or similar languages such as BASIC, flow of control is governed entirely by the programmer; the program goes to the next statement unless told to branch elsewhere. In SAS there is a major exception to this: the usual data step ends in an implicit OUTPUT;RETURN; sequence. This sequence is inserted in the code, as it were, by the SAS Supervisor, but the programmer never sees it. SAS outputs a line to the current SAS data set, and returns control back to beginning of the data step for further executions.

Users may prevent the implicit OUTPUT and RETURN statements from taking effect by putting a STOP statement in the flow of control before the end of the data step is reached.

When the OUTPUT AND/or RETURN statements are desired, programmers learning SAS may find it helpful to explicitly include them, rather than letting the SAS Supervisor insert them implicitly, to better follow the flow of control within their code.

Example Program. The program shown below uses the NULL data set name to prevent a SAS data set from being created. It also uses a STOP statement to prevent the implied OUTPUT;RETURN; sequence from being executed.

```
DATA=null; *NO SAS DATASET NEEDED;
ARRAY mat[6];
ARRAY rowntot[10];
ARRAY colntot[6];
*READ THE INPUT DATA, NUMBER OF ROWS AND COLUMNS ARE;
*ON THE FIRST "CARD," IN FIELDS 1-6 AND 6-10, THEN THE
*DATA MATRIX FOLLOWS, IN 3-COLUMN FIELDS;
INPUT row 1-5,ncol 6-10;
*READ NUMBER OF ROWS AND COLUMNS;
DO row=1 to nrow;
  DO col=1 to ncol;
    INPUT mat[row,col];
    *HOLD RUN FOR MORE READS;
  END;
  *RELEASE RUN;
  STOP; *USE STOP TO AVOID SAS' IMPLIED "OUTPUT" AND "RETURN";
*COMPUTE THE ROW AND COLUMN TOTALS AND THE GRAND TOTAL;
DO row=1 to nrow;
  DO col=1 to ncol;
    rowtot[row]=rowtot[row]+column[row,col];
    colntot[col]=colntot[col]+column[row,col];
  END;
*PRINT THE INPUT MATRIX, WITH HEADINGS AND TOTALS: USE A
*TRAILING @ TO HOLD LINES UNTIL THEY ARE COMPLETED;
FILE PRINT;
PUT row=1, rowntot=8, colntot=8, sum=8;
PUT row=1, rowtot=8, colntot=8, sum=8;
DC COLN=1 TO NCOL;
PUT mat[row,col] 5. @;
DC ROW=1 TO NROW;
PUT rowtot[ROW]=8, colntot=8, sum=8;
DC ROW=1 TO NROW;
STOP; USE "STOP" TO AVOID "STOP" IMPLIED OUTPUT AND RETURN;
*COLUMNS;
10 6
3 5 2 5 0 9
1 9 8 3 1 2
4 2 3 5 0 8
1 7 1 0 0 2
5 1 4 9 0 3
9 8 1 2 3 5
2 2 5 7 1 8
6 9 9 1 4 9
5 1 7 0 1 3
3 8 6 2 5 1
```

The purpose of the example program is to read a matrix of data from cards and print out the data along with row and column totals. Instead of saving the data in a SAS data set, data values are explicitly read into a two-dimensional array.

Typically a SAS program would read the values from a line of input into the program data vector, output it to a SAS data set, and return to the top of the data step to read the next line, until all lines had been read. However in this example, all data lines are read into an array instead, on the first and only cycle through the data step. There is no OUTPUT and no RETURN because STOP halts execution before the end of the data step is reached.

While the code is in SAS, the style is more similar to Fortran than SAS coding practices.

Once beginning SAS programmers understand the similarities between the structure of SAS and languages they have used before, they can more easily transfer old skills, and introduce themselves to the additional features and conventions of SAS without being confused by them.
### Demonstrate Fortran/BASIC-Style Use of SAS Arrays

**Input Matrix, with Totals**

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
<th>COL 4</th>
<th>COL 5</th>
<th>COL 6</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>11</td>
<td>8</td>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>28</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>37</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>SUM</td>
<td>38</td>
<td>51</td>
<td>51</td>
<td>50</td>
<td>41</td>
<td>81</td>
<td>313</td>
</tr>
</tbody>
</table>

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA.