

Bill Fehlner Con't

Partner/Family: Married for 36 years, has 4 adult daughters and 2 grandchildren.

When I'm not at SAS, I ... enjoy wilderness canoe camping, using the cedar strip canoe that one of my daughters and I recently built. Music is also a big part of my life, either listening or playing my trumpet for personal enjoyment. Cross country skiing in the winter and cycling in the summer keep me in shape for my canoe trips.

SAS 'Techie Tip': When using a SAS data step to locate information, there are situations where we want to stop a search as soon as we find a result. From the early days, the SAS data step has included the DO WHILE statement and the DO UNTIL statement. But in the current version of SAS, one can also use the LEAVE; statement.

For example, a SAS data set containing medical records has 16 diagnostic variables. We want to locate all observations with at least one occurrence of diagnostic value 7742 . The following code does it but doesn't know when to stop.

```
Data work.results(drop = loop found);
  set medicalRecords;
  array diagnostics (*) diag1-diag16;
  found = 0;
  do loop = 1 to 16;
    if (diagnostics(loop) = '7742' then do;
      found = 1;
    end;
  end;
  if found then output;
run;
```

The next example uses DO WHILE. It knows when to stop, but requires some careful coding in the logical expression.

```
Data work.results(drop = loop found);
  set medicalRecords;
  array diagnostics (*) diag1-diag16;
  found = 0;
  do loop = 1 to 16 while(found = 0);
    if (diagnostics(loop) = '7742' then do;
      found = 1;
    end;
  end;
  if found then output;
```

```
run;
```

The final example uses LEAVE. It knows when to stop, and only requires that you can spell LEAVE correctly.

```
Data work.results(drop = loop found);
  set medicalRecords;
  array diagnostics (*) diag1-diag16;
  found = 0;
  do loop = 1 to 16;
    if (diagnostics(loop) = '7742' then do;
      found = 1;
      LEAVE;
    end;
  end;
  if found then output;
run;
```

DO WHILE always checks a single logical condition at the bottom on the DO loop. DO UNTIL checks a single logical condition at the top of the DO loop. LEAVE can be invoked anywhere, top, bottom or middle. Because LEAVE can appear multiple times within a single DO loop, you do not have to try to combine multiple conditions into a single, complex logical expression which can be difficult to debug.