

## Beyond IF THEN ELSE: Techniques for Conditional Execution of SAS Code

Joshua M. Horstman, Nested Loop Consulting, Indianapolis, IN

### ABSTRACT

Nearly every SAS program includes logic that causes certain code to be executed only when specific conditions are met. This is commonly done using the IF...THEN...ELSE syntax. In this paper, we will explore various ways to construct conditional SAS logic, including some that may provide advantages over the IF statement. Topics will include the SELECT statement, the IFC and IFN functions, the COALESCE function, as well as some more esoteric methods, and we'll make sure we understand the difference between a regular IF and the %IF statement in the macro language.

### INTRODUCTION

The conditional execution of code is one of the most fundamental concepts in computer programming of all types. The ability to take different actions based on different inputs is essential in order to complete complex tasks. Indeed, most SAS programs are chock full of conditional logic.

The most common method for implementing conditional logic in SAS, and probably the first learned by most SAS programmers, is the IF...THEN...ELSE statement. This construct provides a simple means of branching the flow of execution based on whether a specified condition is true or false.

However, there are many other techniques available to the SAS programmer. We will examine several of them in this paper, study examples of their use, and consider what advantages they may offer over the IF...THEN...ELSE statement.

### THE SELECT STATEMENT

One alternative to the IF...THEN...ELSE statement is the SELECT statement. The SELECT statement is always used in conjunction with a group of one or more WHEN statements. This structure can operate in one of two ways.

The first and most common way to use the SELECT statement is to specify an expression as part of the SELECT statement. That expression is compared to the expression(s) associated with each WHEN statement. When the SELECT expression is equal to one of the WHEN expressions, an associated statement is executed. A WHEN statement can have more than one WHEN expression. An optional OTHERWISE statement can specify a statement to be executed only if none of the WHEN expressions match the SELECT expression.

Consider the following example demonstrating use of the SELECT statement:

```
select(customer_type);
  when('STANDARD') total = price * taxrate;
  when('PREFERRED') total = price * discount * taxrate;
  when('TAXEXEMPT') total = price;
  when('GOVERNMENT') total = price * discount;
end;
```

The exact same logic can also be implemented using the IF...THEN...ELSE statement:

```
if customer_type = 'STANDARD' then total = price * taxrate;
else if customer_type = 'PREFERRED' then total = price * discount * taxrate;
else if customer_type = 'TAXEXEMPT' then total = price;
else if customer_type = 'GOVERNMENT' then total = price * discount;
```

While these two methods produce the same result, the structure of the SELECT statement is more readily apparent at a glance, making it easier to read, debug, modify, and reuse.

## THE IFC AND IFN FUNCTIONS

Another way to implement conditional logic is through the use of the IFC and IFN functions, which were new in SAS version 9. In certain situations, these functions can produce a much more compact and elegant alternative to IF...THEN...ELSE logic.

Both functions accept the same four arguments: a logical expression to be tested, a value to be returned if the expression is true, a value to be returned if the expression is false, and, optionally, a value to be returned if the expression is missing. The difference between the two functions is that IFC returns a character value while IFN returns a numeric value.

Consider the following alternative to the first example shown above.

```
total = price *
      ifn(customer_type in ('PREFERRED', 'GOVERNMENT'), discount, 1) *
      ifn(customer_type in ('STANDARD', 'PREFERRED'), taxrate, 1);
```

This single assignment statement replaces an entire block of IF...THEN...ELSE logic, while simultaneously making the underlying logic behind the derivation of TOTAL more clear. In each of these IFN functions, if the expression is false, we have cleverly returned a value of 1, which will have no effect when used as a factor in multiplication.

The IFC function can be particularly useful when building strings constituting of multiple components determined by various pieces of conditional logic. Observe the IF...THEN...ELSE logic in the following example.

```
if married='Y' and num_kids=0 then family_status = 'Married, no children';
if married='N' and num_kids=0 then family_status = 'Unmarried, no children';
if married='Y' and num_kids=1 then family_status = 'Married, 1 child';
if married='N' and num_kids=1 then family_status = 'Unmarried, 1 child';
if married='Y' and num_kids>1 then family_status = 'Married, ' ||
strip(put(num_kids,best.)) || ' children';
if married='N' and num_kids>1 then family_status = 'Unmarried, ' ||
strip(put(num_kids,best.)) || ' children';
```

The IFC function can be used as a convenient way to simplify the logic and make it easier to change or reuse in the future.

```
family_status = catx(' ', ifc(married='Y', 'Married, ', 'Unmarried, '),
                    ifc(num_kids=0, 'no', put(num_kids,best.)),
                    ifc(num_kids=1, 'child', 'children'));
```

## THE COALESCE AND COALESCEC FUNCTIONS

The COALESCE and COALESCEC functions can be used to implement conditional logic when the purpose of the logic is to select the first non-missing value among several expressions. That is, in fact, precisely what these functions do. Note that COALESCE is used with numeric values while COALESCEC operates on character data.

Consider the following example. We wish to derive the last date of contact (LSTCONDT) for each subject in a clinical trial. If the subject has died, then the date of death (DTHDT) is used. If DTHDT is missing, then the subject has not died. If the subject is alive but has withdrawn from the study, the date of withdrawal (WDDT) will be used. If WDDT is missing, then the subject has not withdrawn. If the subject is still on study, then the date of the last dose (LSTDOSDT) is used. If LSTDOSDT is missing, then the subject has not been dosed and we will instead use the date of the last visit (LSTVISDT).

The logic described above can be implemented using the following IF...THEN...ELSE syntax.

```
if not missing(dthdt) then lstcondt = dthdt;
else if not missing(wddt) then lstcondt = wddt;
else if not missing(lstvisdt) then lstcondt = lstdosdt;
else lstcondt = lstvisdt;
```

Observe how the COALESCE function allows us to condense the above logic into a single, succinct statement.

```
lstcondt = coalesce(dthdt, wddt, lstdosdt, lstvisdt);
```

## THE SUBSETTING IF

Sometimes a SAS programmer will need to use conditional logic to exclude or subset records. In such cases, the subsetting IF statement can be useful. The syntax of the subsetting IF statement different from that of the ordinary IF...THEN...ELSE statement in that there is neither a THEN or an ELSE. Rather, the statement consists solely of an IF statement followed by an expression to be evaluated. If the expression evaluates to true, data step processing continues and the current record is written to the output data set. If not, the current iteration of the DATA step terminates without writing any output and control returns to the top of the DATA step.

The subsetting IF statement is functionally equivalent to an IF...THEN statement written in the following manner.

```
if <expression> then
  do;
    <all subsequent DATA step statements>
  output;
end;
```

## THE %IF MACRO STATEMENT

One thing that is frequently confusing to new SAS programmers is the difference between the IF...THEN...ELSE statement and the %IF...%THEN...%ELSE macro statement. While these two are similar in their logical construction, they operate at entirely separate levels within a SAS program. The IF...THEN...ELSE statement is a DATA step statement and can only be used therein. The %IF...%THEN...%END macro statements are actually part of the SAS/Macro facility, which is a separate language with its own syntax that can be used to generate SAS code.

The %IF macro statement allows you to conditionally generate SAS code, usually based on the value of one or more macro variables. The SAS code generated might be DATA step code, SAS procedures, or portions or combinations thereof. It could even be conditional code like that described previously.

## CONCLUSION

There are many ways to implement conditional logic with SAS. One method is not necessarily better than another. Programmers should strive to write code that is simple, intuitive, and easy to modify, extend, or reuse. Sometimes it will not be possible to satisfy all of these criteria, so professional judgment must be exercised. Savvy SAS programmers will still use the IF...THEN...ELSE statement frequently, perhaps even primarily, but they also benefit from having a variety of tools available and knowing how and when to use them.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Joshua M. Horstman  
Nested Loop Consulting  
8921 Nora Woods Drive  
Indianapolis, Indiana 46240  
317-414-9584  
josh@nestedloopconsulting.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.