Learning SAS: Mastering String Manipulation with the FINDC Function

Authored by

Mohammed looti

November 14, 2025

In the vast landscape of data processing and analysis, especially within the environment of SAS programming, the mastery of text processing is paramount. A core requirement for data professionals is the ability to execute precise and reliable string manipulation operations. Among the suite of powerful SAS functions dedicated to text searching, the **FINDC** function holds a unique and highly specialized position. This function is engineered specifically for character-level searching, enabling users to accurately identify the starting position of the first occurrence of any single character that belongs to a designated set within a target string. Its distinction lies in its focus: while many other functions search for entire sequences or words, **FINDC** is deliberately optimized for rapid identification of individual characters defined by the user.

The utility of the **FINDC** function spans critical areas of data governance and transformation. It is an indispensable tool for tasks such as validating incoming data fields, efficiently identifying the presence of restricted or special characters, and generally preparing unstructured textual data for rigorous analysis. By providing the exact, 1-based index location of a matching character from a defined list, **FINDC** empowers programmers to execute targeted cleaning, truncation, or modification of text variables with high fidelity. Proficiency in leveraging the capabilities of **FINDC** significantly elevates a programmer's capacity to manage complex textual datasets, offering a degree of granular control that is unattainable with less specialized search routines.

Deconstructing the FINDC Function Syntax

The syntax governing the operation of the **FINDC** function is designed for maximum clarity and operational efficiency. Unlike functions requiring numerous optional parameters, **FINDC** requires only two mandatory arguments to successfully perform its comprehensive character search. This straightforward structure makes it highly accessible for new SAS users while maintaining the robustness needed for complex production tasks.

The fundamental structure defining its usage is elegantly simple:

FINDC(string, charlist)

Accurate implementation hinges on a clear understanding of the role played by each component in the search mechanism:

string: This constitutes the primary input--the target character variable or literal value that the function will systematically scan. The search operation executes sequentially, starting at the first position and proceeding until a match is confirmed or the string concludes.
charlist: This critical argument specifies the collective set of individual characters that the function is instructed to locate within the provided *string*. It is essential to internalize that **FINDC** seeks the first occurrence of *any* single character contained within this list, rather than looking for a specific pattern or sequence. If multiple matching characters from the *charlist* are present in the *string*, the

function is programmed to reliably return the position of the very first one encountered during its standard left-to-right scan.

Crucially, the result generated by the **FINDC** function is always an integer value. This value precisely corresponds to the 1-based starting position of the initial matching character found. If the function completes its scan of the entire *string* and fails to detect any of the characters specified in the *charlist*, it consistently returns the distinct control value of **0**. This zero return value is a powerful mechanism for conditional processing and subsequent data filtering within larger SAS programming routines.

Practical Application: Setting Up the Demonstration Data

To fully appreciate the practical utility and specific behavior of the **FINDC** function, we will proceed through a detailed, hands-on example. This scenario simulates a common real-world data processing requirement: analyzing a list of records to identify specific character occurrences within textual variables. Our objective is to clearly demonstrate how **FINDC** seamlessly integrates into a standard SAS workflow to extract targeted insights from text.

We initiate the demonstration by constructing a sample SAS dataset named `original_data`. We use a foundational DATA step to populate this dataset, which contains a single character column, `name`, populated with several full names. This controlled environment ensures that we can precisely observe and verify the function's output behavior against known inputs, providing a clear foundation for understanding its mechanics.

```
/*create dataset*/
data original_data;
input name $25.;
datalines;
Andy Lincoln Bernard
Barren Michael Smith
Chad Simpson Arnolds
Derrick Smith Henrys
Eric Millerton Smith
Frank Giovanni Goode
;
run;

/*view dataset*/
proc print data=original_data;
```

The execution of this block of code successfully generates our foundational dataset. We utilize PROC PRINT to confirm the structure, which consists of a simple table containing the names that will serve as the target strings for the subsequent character search operations. The visual confirmation below validates the integrity of the data structure prior to any processing steps.

Obs	name
1	Andy Lincoln Bernard
2	Barren Michael Smith
3	Chad Simpson Arnolds
4	Derrick Smith Henrys
5	Eric Millerton Smith
6	Frank Giovanni Goode

Implementing FINDC for Targeted Character Search

With our preparatory data now secured, we can proceed to demonstrate the core functionality of the **FINDC** function in a practical context. For this specific scenario, our objective is highly targeted: we aim to locate the position of the first occurrence of any character within the set **'x', 'y', or 'z'** within each entry of the `name` column. This type of search is often employed in data cleansing protocols where analysts need to flag records containing unusual, non-standard, or restricted characters.

To achieve this, we define a new dataset, `new_data`, which inherits all existing variables from `original_data`. Crucially, we introduce a new variable, `first_xyz`, whose value is calculated by applying the **FINDC** function. The code segment below showcases the integration of **FINDC** directly within the SAS DATA step, where the target string (`name`) and the specific character list (`'xyz'`) are explicitly defined as arguments.

```
/*find position of first occurrence of either x, y or z in name*/
data new_data;
set original_data;
first_xyz = findc(name, 'xyz');
run;

/*view results*/
proc print data=new_data;
```

The resulting output, generated via another execution of PROC PRINT, clearly displays the original

names alongside the newly calculated `first_xyz` values. These integer values precisely map to the 1-based index where the earliest matching character from the specified 'x', 'y', 'z' set was successfully located within the respective name string.

Obs	name	first_xyz
1	Andy Lincoln Bernard	4
2	Barren Michael Smith	0
3	Chad Simpson Arnolds	0
4	Derrick Smith Henrys	19
5	Eric Millerton Smith	0
6	Frank Giovanni Goode	0

A detailed analysis of the calculated `first_xyz` results confirms the operational logic of **FINDC**:

When processing "Andy Lincoln Bernard", the character 'y' is the first match encountered, found at the 4th position, correctly yielding a return value of **4**.
For "Barren Michael Smith", a comprehensive scan reveals that none of the characters 'x', 'y', or 'z' are present anywhere in the string. Consequently, the function adheres to its protocol and returns the definitive value of **0**.
In the case of "Derrick Smith Henrys", the character 'y' is eventually located within the final segment ("Henrys") at position 16, resulting in a recorded value of **16**.

This consistency, particularly the reliable return of **0** when no matching character is found, is highly advantageous in data processing workflows. This feature simplifies conditional logic, allowing programmers to effortlessly filter or subset the dataset to isolate records that either require further scrutiny or confirm compliance based on the absence of certain characters.

Distinguishing FINDC from the FIND Function

While both **FINDC** and the related FIND function serve as foundational tools for searching within strings in SAS, they operate under fundamentally different mechanisms and, therefore, fulfill distinct purposes. Selecting the correct function for a given task is essential for developing efficient, accurate, and logically sound SAS programming code that performs as intended.

The core disparity lies in their search criteria: the FIND function is strictly designed to locate the starting position of the first occurrence of a complete, contiguous substring. It demands an exact, sequential match of the search term. In direct contrast, the **FINDC** function, as we have demonstrated, searches for the first instance of *any* individual character specified within its

character list argument, operating without any requirement for sequence or contiguity. This difference makes **FINDC** ideal for character validation and **FIND** ideal for phrase identification.

To vividly highlight this crucial operational disparity, we will execute a direct comparison using the search term 'Smith' against our sample data. We will augment our dataset by introducing two new variables: `find_smith` (calculated using the FIND function) and `findc_smith` (calculated using the **FINDC** function). Observing the resulting indices side-by-side will clearly illustrate how each function interprets the identical search query based on its underlying search mechanism.

```
/*create new dataset*/
data new_data;
set original_data;
find_smith = find(name, 'Smith');
findc_smith = findc(name, 'Smith');
run;

/*view new dataset*/
proc print data=new_data;
```

The resulting output below provides a compelling visualization of the functional difference when searching the same data using both methods. Pay close attention to the dramatic divergence in index values displayed in the `find_smith` and `findc_smith` columns.

Obs	name	find_smith	findc_smith
1	Andy Lincoln Bernard	0	7
2	Barren Michael Smith	16	9
3	Chad Simpson Arnolds	0	2
4	Derrick Smith Henrys	9	5
5	Eric Millerton Smith	16	3
6	Frank Giovanni Goode	0	8

In the `find_smith` column, the function successfully locates the exact, contiguous substring 'Smith'. For instance, in "Barren Michael Smith," the word begins at position 16, resulting in a value of **16**. However, for "Andy Lincoln Bernard," since the complete sequence 'Smith' is entirely absent, `find_smith` correctly returns **0**. Conversely, the `findc_smith` column interprets 'Smith' not as a word, but as a collection of five independent characters: ('S', 'm', 'i', 't', 'h'). When processing "Andy

Lincoln Bernard," the character 'i' (which is part of the character list) is the first match found, occurring at position 7 (within the word "Lincoln"), leading `findc_smith` to return **7**. Similarly, for "Barren Michael Smith," the character 'm' is found first at position 8 (in "Michael"), returning **8**. This powerful comparison decisively confirms that **FINDC** is a character-level, set-based tool, while **FIND** is a phrase-level tool designed for locating continuous sequences within a string.

Conclusion and Best Practices for String Searching

The **FINDC** function represents an indispensable resource within the SAS programming arsenal for conducting highly precise, character-level string manipulation. Its design makes it uniquely suited for scenarios that require the swift detection and accurate location of individual characters derived from a predefined set. This capability offers significant advantages for essential tasks such as data validation, rigorous data cleansing, and the identification of complex or irregular patterns embedded within textual variables.

When approaching any SAS string search task, adopting a critical selection rule is paramount for optimizing code performance and accuracy: utilize **FINDC** specifically when your objective is to pinpoint the first position of any single character from a user-defined collection. Conversely, you should reserve the FIND function exclusively for identifying the starting index of an entire, consecutive substring. Making this judicious choice ensures that your SAS programs are both logically robust and highly performant across diverse data volumes.

We strongly encourage further exploration, urging users to experiment with various character lists, including special characters and control codes, against complex strings. Fully understanding the nuances of **FINDC** allows developers to harness its full power in dynamic and challenging data processing workflows, ultimately leading to higher quality and more reliable data output.

Additional Resources

The following tutorials explain how to use other common functions in SAS: