How to use the INDEXW function in SAS?

Authored by

stats writer

November 19, 2025

The INDEXW function in SAS is an indispensable tool for advanced string manipulation and precise word searching within character variables. Unlike basic character-searching functions, **INDEXW** is specifically designed to locate the position of a complete word, rather than just a sequence of characters, making it highly valuable for text processing and natural language tasks within the analytic environment. This powerful function helps analysts determine the exact starting position of a targeted word within a source variable, returning a numeric value that represents that starting point. If the specified word is not found, the function gracefully returns a value of zero.

Effective data cleaning and feature engineering often require the ability to isolate specific textual components. When working with large volumes of unstructured or semi-structured text data, relying on word boundaries is essential to prevent false positives that can arise from searching for substrings embedded within other words. The primary benefit of employing the INDEXW function lies in its efficiency and accuracy in contextual searches, allowing users to quickly locate and flag records based on the presence or absence of specific vocabulary. This saves significant computational resources compared to manual searching or complex regular expression patterns for simple word identification.

Understanding how **INDEXW** operates is fundamental for any SAS programmer tasked with complex data transformations. It requires defining two essential components: the source character variable being analyzed and the target word being sought. The function then processes the source variable, respecting defined delimiters (usually spaces, but customizable through optional arguments) to ensure that only standalone words match the search criteria. The resulting numeric output facilitates subsequent conditional processing, variable creation, or filtering operations within the SAS Data Step.

Understanding the INDEXW Function Syntax

The structure required to implement the **INDEXW** function is straightforward, yet precise. Mastery of the correct syntax ensures the function performs the intended word search correctly across your data set. The basic form focuses on defining the input text and the target word, but for more specialized applications, optional arguments can be utilized to customize the search behavior, particularly regarding case sensitivity and delimiter recognition. However, for most standard applications, the two mandatory arguments suffice for accurate word location.

The basic syntax for utilizing this function is as follows:

INDEXW(source, excerpt)

where:

source: This argument specifies the primary character expression or variable that the function will

scan. This is the complete string within which the word search takes place.
excerpt: This argument represents the specific word you intend to locate within the *source* string. It must be enclosed in quotation marks if it is a literal value, or it can be a character variable containing the word to be searched.

The function returns a numeric integer. A positive integer indicates the position of the first character of the located word. A return value of **1** means the word starts at the very beginning of the source string. Conversely, a return value of **0** signals that the specified word was not present in the source variable, respecting the definition of a word (i.e., bounded by delimiters like spaces). This zero return value is critical for filtering or setting flags in data quality checks.

Example 1: Using the INDEXW Function for Word Location

To demonstrate the utility of the INDEXW function, let us consider a practical scenario involving a small data set containing various phrases. Our goal is to locate the position of the word 'pig' within each phrase. This example highlights how **INDEXW** correctly identifies word boundaries, distinguishing standalone words from character sequences that might be part of longer terms.

We begin by constructing a sample data set in the SAS environment using the Data Step. This data includes a single character column named **phrase**, populated with diverse sentences related to the term 'pig'. Notice the inclusion of phrases where 'pig' appears as a full word, and phrases where 'pig' is part of a larger word (like 'piglet' or 'piggie').

```
/*create dataset: Defining the source phrases*/
data original_data;
input phrase $40.;
datalines;
A pig is my favorite animal
My name is piglet
Pigs are so cute
Here is a baby pig
His name is piggie
;
run;

/*view dataset: Displaying the newly created raw data*/
proc print data=original_data;
```

Obs	phrase
1	A pig is my favorite animal
2	My name is piglet
3	Pigs are so cute
4	Here is a baby pig
5	His name is piggie

Executing the Search and Analyzing Positional Output

We now execute the word search using the INDEXW function. In the new Data Step, we create a variable named **indexw_pig**. This variable is assigned the result of the **INDEXW** operation, utilizing **phrase** as the source string and 'pig' as the target word. The code below illustrates this transformation, followed by a procedural step to display the results.

```
/*find position of first occurrence of 'pig' in phrase column, respecting word boundaries*/
data new_data;
set original_data;
indexw_pig = indexw(phrase, 'pig');
run;

/*view results: Displaying the original phrase and the calculated position*/
proc print data=new_data;
```

Obs	phrase	indexw_pig
1	A pig is my favorite animal	3
2	My name is piglet	0
3	Pigs are so cute	0
4	Here is a baby pig	16
5	His name is piggie	0

The resulting data set, **new_data**, now contains the variable **indexw_pig**, which displays the precise starting position of the word 'pig'. By examining the output, we can observe crucial differences based on the context of the word. For example, in the first observation ('A pig is my favorite animal'), the word 'pig' starts at position **3**. Crucially, **INDEXW** correctly returns a position

for observations 1, 3, and 4 where 'pig' appears as a distinct word.

A key observation is the handling of phrases where 'pig' is only present as a root component of a longer word, such as 'piglet' or 'piggie' (observations 2 and 5). In these cases, since **INDEXW** enforces strict word boundary rules, the function returns a value of **0**. This distinct behavior confirms that the function prioritizes finding the standalone word, making it superior to general character searches when linguistic specificity is required. If the word being searched for never occurs in the **phrase** column as a complete word, the **INDEXW** function consistently returns **0**, providing a clear binary indicator of presence or absence.

The Critical Distinction Between INDEXW and INDEX Functions

While the INDEXW function is specialized for finding whole words, INDEX (without the 'W') is the standard SAS function used for locating any arbitrary sequence of characters, known as a substring. Understanding this fundamental difference is vital for selecting the correct tool for your text analysis task. Using **INDEX** when you require word-level precision can lead to inaccurate results, as it treats all character sequences equally, regardless of surrounding delimiters.

The **INDEX** function searches the source string and returns the position of the first occurrence of the target substring, even if that substring is embedded within a larger word. For instance, searching for 'car' using **INDEX** would successfully find it within the word 'carpet'. By contrast, the **INDEXW** function mandates that the target sequence must be clearly delineated by specified word boundaries (like spaces, punctuation, etc.) to be counted as a 'word'. By default, a **word** in the context of **INDEXW** must be flanked by delimiters or be at the start/end of the source string.

This difference in definition is the primary source of operational contrast. If your objective is to find any partial match--for example, identifying records containing a specific product code fragment--you would use **INDEX**. However, if your objective is linguistic analysis--for example, counting documents that mention "cloud computing" as a dedicated term--then **INDEXW** is the appropriate, more precise choice. Misapplication of these functions is a common error in text data processing, leading to flawed analytical outcomes.

Illustrating Positional Differences Using Both Functions

To definitively illustrate the operational differences between **INDEX** and INDEXW function, we will modify our previous Data Step to calculate both functions simultaneously on the same input data. We will create two new variables: **index_pig** (using the **INDEX** function) and **indexw_pig** (using the **INDEXW** function), both searching for the sequence 'pig'.

The following SAS code block shows the parallel application of both functions, followed by the printing procedure. Observe carefully how the results diverge, particularly in observations 2 and 5,

where 'pig' is part of 'piglet' or 'piggie'.

```
/*create new dataset: Calculating both INDEX and INDEXW outputs*/
data new_data;
set original_data;
index_pig = index(phrase, 'pig');
indexw_pig = indexw(phrase, 'pig');
run;

/*view new dataset: Comparing the results side-by-side*/
proc print data=new_data;
```

Obs	phrase	index_pig	indexw_pig
1	A pig is my favorite animal	3	3
2	My name is piglet	12	0
3	Pigs are so cute	0	0
4	Here is a baby pig	16	16
5	His name is piggie	13	0

The output table provides a clear side-by-side comparison. The **index_pig** column displays the position of the first occurrence of the substring 'pig' in the **phrase** column. Notice that for every single observation, **index_pig** returns a positive number (3, 11, 1, 15, 12), because the character sequence 'pig' exists in every phrase, even if embedded within 'piglet' or 'piggie'.

In stark contrast, the **indexw_pig** column displays the position of the first occurrence of the **word** 'pig'. For observations 2 ('My name is piglet') and 5 ('His name is piggie'), **indexw_pig** returns **0**. This zero value confirms that, although the characters 'p-i-g' are present, they are not surrounded by word delimiters; thus, they are not recognized as the discrete word 'pig'. This example is the definitive proof of why **INDEXW** is essential for word-level precision in SAS data processing.

Advanced Usage: Controlling Delimiters in INDEXW

While the basic implementation of **INDEXW** relies on standard delimiters (like space, period, comma, etc.), SAS allows for the customization of these word separators through optional arguments. This advanced functionality is necessary when dealing with data where words might be separated by non-standard characters, such as underscores, hyphens, or custom symbols, which must be treated as word boundaries.

The full syntax of the **INDEXW** function includes an optional third argument, known as the **delimiters** argument: `INDEXW(source, excerpt, delimiters)`. If this argument is supplied, the function will use only the characters provided in that argument as word separators. For example, if you set the delimiters to be only a hyphen ('-'), then a space (' ') would no longer be recognized as a word boundary, drastically altering the search results.

The ability to define custom delimiters offers substantial flexibility for parsing specialized data formats, such as log files, coded strings, or concatenated identifiers where standard spaces are absent. By defining exactly what constitutes a word boundary, analysts gain precise control over how text is tokenized, ensuring that **INDEXW** accurately identifies logical units even in complex character fields. Always consult the official INDEXW function documentation when implementing custom delimiters to understand default behavior and optional modifiers.

Summary of Word Searching Functions in SAS

The INDEXW function is a critical component of the SAS toolkit for anyone engaging in rigorous text analysis or data validation. Its specialized ability to locate the starting position of a whole word, defined by recognizable boundaries, makes it distinctly valuable compared to the more general **INDEX** function, which only searches for substrings.

When preparing data for machine learning models or detailed reporting, ensuring linguistic accuracy is paramount. Using **INDEXW** prevents spurious matches and provides confidence that the analytical results are based on the presence of complete, meaningful tokens within the text. If your work involves filtering text based on vocabulary, or calculating the complexity of phrases based on word location, **INDEXW** should be your function of choice.

For users seeking further mastery of character manipulation in SAS, exploring related functions such as **FINDW** (similar to INDEXW but returns true/false), **SCAN** (for extracting words based on delimiters), and **COMPBL** (for collapsing multiple spaces) can significantly enhance your data processing capabilities. These tools collectively provide a robust framework for handling virtually any character variable challenge encountered in large-scale data environments.