Learning to Find Words in SAS: A Guide to the INDEXW Function

Authored by

Mohammed looti

November 14, 2025

In the highly analytical and rigorous environment of SAS programming, the ability to expertly manage and analyze textual information is paramount. Effective handling of character strings is not merely a beneficial skill but a fundamental requirement for success in data science. Tasks such as comprehensive data cleaning, precise information extraction, and complex text mining workflows are all built upon the foundation of accurate text manipulation. To execute these tasks with precision, SAS developers must possess a deep understanding of the extensive suite of built-in functions available within the system. Among the most critical tools for textual analysis stands the **INDEXW** function, which offers unmatched capability for locating specific, whole words within larger text segments.

The **INDEXW** function in SAS is a specialized utility specifically designed to identify the starting position of the first character of a complete, standalone word found within a given source string. This specialized functionality is crucial because it resolves a common ambiguity in text processing: the need to confirm that search criteria match full, discrete words, rather than simply identifying sequential character patterns or substrings embedded within other words. By strictly enforcing word boundaries, **INDEXW** drastically improves the reliability and accuracy of text analysis results, enabling much more precise filtering, extraction, and data manipulation.

This comprehensive guide is designed to thoroughly dissect the mechanics, syntax, and real-world applications of the **INDEXW** function. We will walk through clear, step-by-step examples using executable SAS code and provide a rigorous comparison of **INDEXW** against its closely related counterpart, the standard **INDEX** function. By the conclusion of this article, you will be fully equipped with the technical expertise necessary to effectively leverage the power and precision of **INDEXW** in all your daily analytical and text processing duties.

Core Concepts of the INDEXW Function in SAS

The **INDEXW** function serves as a highly specialized search utility engineered for performing exact, word-specific lookups within character variables in the SAS environment. Its primary distinguishing feature is its unwavering commitment to respecting word boundaries, setting it apart from generic character search methods. Unlike simple search functions that merely scan for any sequence of characters, **INDEXW** rigorously demands that the search term must exist as an entire, properly delimited word. This feature is indispensable for accurate semantic text processing, as it systematically eliminates the common risk of obtaining false positives--results where a search term is incorrectly identified because it is merely a segment embedded within a longer, unrelated word.

Implementing the **INDEXW** function is simplified by its concise and logical syntax. While the function supports optional parameters for more granular control over the search, the basic usage requires the specification of two fundamental arguments:

INDEXW(source, excerpt)

A detailed explanation of the essential, mandatory parameters provides clarity on the core components required to successfully execute a precise word search operation:

source: This argument defines the primary character string or variable that is to be subjected to analysis. It represents the complete body of text where the search operation will be conducted.
excerpt: This parameter specifies the exact word that the function is tasked with locating within the designated *source* string. The function will perform a strict search, demanding an exact, standalone match of this word, meticulously adhering to all defined word boundaries.

Upon execution, the function produces an integer value corresponding to the starting column position of the first character of the identified word. Crucially, if the specified word is entirely absent from the *source* string--meaning no standalone, whole-word match is successfully identified--**INDEXW** is guaranteed to return a value of **0**. This distinct, standardized result is extremely valuable for programmers, as it allows for straightforward integration of the function into complex conditional logic used for efficient filtering, categorization, or flagging of observations based on the confirmed presence or absence of specific keywords.

Practical Implementation: Setting Up the SAS Data Step

To fully grasp the practical utility and inherent precision of the **INDEXW** function, we will proceed with a concrete, working example. Our immediate goal in this scenario is to pinpoint the exact starting position of a particular target word within a small collection of descriptive phrases. This task accurately mirrors real-world challenges encountered in text analytics, where the successful location of specific keywords is a vital precursor to almost all subsequent data processing and analytical steps.

We begin the demonstration by constructing a simple, foundational SAS dataset using the standard DATA step procedure. This newly created dataset, which we name `original_data`, will contain a single character column titled `phrase`. This column is populated with various sample sentences, each containing variations or instances of the search target word, "pig":

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

Following the successful execution of this SAS code block, the system generates the initial structured dataset, which is now properly prepared for analytical manipulation. The output confirms the structure:

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

This `original_data` serves as the robust starting point for our detailed functional demonstration. Since each row represents a distinct character string, we are now ideally positioned to apply the specialized logic of the **INDEXW** function. Our next phase involves systematically processing these phrases to search specifically for our target whole word, 'pig', while ignoring any partial matches.

Applying INDEXW to Locate Whole Words

The subsequent logical step in our demonstration involves the effective implementation of the **INDEXW** function. We will utilize it to search exclusively for the whole word 'pig' within every existing entry of the `phrase` column in our previously constructed `original_data`. To store and analyze the results, we will construct a new SAS dataset, named `new_data`, and introduce a newly derived variable, `indexw_pig`. This variable will be dedicated to housing the computed positional results generated by our targeted word search.

The following SAS code block clearly illustrates the straightforward application of the function within a new DATA step. This structure is typical for analytical tasks where new calculated variables must be added to an existing dataset:

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

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

After executing this code, the resulting `new_data` dataset is rendered, providing a clear visual representation of the newly calculated column, `indexw_pig`, which is now populated with the precise positional results derived from the search operation:

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 output provides compelling evidence of the highly precise nature of **INDEXW**. In the first phrase, "A pig is my favorite animal," the function accurately locates the word 'pig' starting at position 3. Most significantly, in the rows containing the phrases "My name is piglet" and "His name is piggie," the function deliberately returns a value of **0**. This result occurs because the target term 'pig' does not exist as a standalone, complete word in those strings; instead, it is merely an embedded component of a longer, different word. This crucial distinction confirms that **INDEXW** successfully and rigorously enforces word boundary recognition, thereby effectively distinguishing between complete lexical units and simple embedded character sequences.

Distinguishing INDEX from INDEXW: Substring vs. Word Search

A core element of effective string manipulation within SAS lies in understanding the critical functional disparities between the standard **INDEX** function and the specialized **INDEXW** function. While both are used to locate patterns of characters, their underlying methodologies diverge fundamentally: **INDEX** is designed for a general substring search, whereas **INDEXW** is strictly dedicated to whole-word searching.

The standard **INDEX** function operates by scanning for and returning the starting position of the first occurrence of any arbitrary sequence of characters--the substring--within a larger string. It functions purely on the basis of character sequence matching, meaning it completely disregards the linguistic context of word boundaries. If we were to use the **INDEX** function to search for 'pig', it would successfully yield a match in words like 'piglet', 'piggie', and 'pigment', because the sequence 'p-i-g' is present, irrespective of whether 'pig' itself constitutes a standalone word.

In powerful contrast, the **INDEXW** function is profoundly context-aware. It strictly adheres to the definition of a "word," which is conventionally delineated by spaces, various punctuation marks, or other defined special characters, collectively known as delimiters. This rigid requirement for boundary recognition ensures that **INDEXW** will only positively identify and return a positional value if the target term exists as a complete, separate, and properly delimited word. This unique capability makes **INDEXW** an indispensable tool for performing true semantic searches, reliable keyword extraction, and any analytical application where precise word separation is an absolute necessity for data accuracy.

To unequivocally illustrate this essential distinction, we will now execute a procedure that creates a new dataset where both the **INDEX** and **INDEXW** functions are applied simultaneously to the exact same set of input phrases. This side-by-side comparison is the most effective way to visually demonstrate how each function interprets and processes the search term 'pig' within our textual data:

```
/*create new dataset*/
data new_data;
set original_data;
index_pig = index(phrase, 'pig');
indexw_pig = indexw(phrase, 'pig');
run;

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

The resulting dataset, which now includes the comparative columns `index_pig` and `indexw_pig`, provides a stark and direct visualization of the two distinct search methodologies:

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

Observe the powerful contrast in the results, particularly in the rows containing "piglet" and "piggie." The `index_pig` column, utilizing the standard **INDEX** function, correctly identifies the partial sequence 'pig' starting at position 1 in both instances. In direct opposition, the `indexw_pig`

column consistently returns **0** for these identical rows because 'pig' fails to meet the criterion of being a standalone, delimited word. This compelling demonstration confirms that the selection between **INDEX** and **INDEXW** must be carefully governed by your specific analytical requirement: whether you need to match a partial character sequence or a complete, boundary-recognized word.

Advanced Usage and Custom Delimiters

While the fundamental application of **INDEXW** is sufficient for standard text processing challenges, mastering its advanced functionalities--especially concerning custom delimiters and case sensitivity--can substantially broaden a SAS developer's capacity for complex data manipulation. By default, **INDEXW** relies on a predefined list of characters, such as spaces, periods, slashes, and parentheses, to establish and recognize word boundaries. This default set handles most conventional prose effectively.

However, when dealing with highly specialized formats or structured text data, analysts frequently need the capability to define entirely custom word boundaries. Fortunately, **INDEXW** is highly accommodating in this regard, supporting an optional third argument that permits the specification of a string containing the user's desired delimiters. The complete syntax becomes `INDEXW(source, excerpt, delimiters)`. This level of flexibility is absolutely invaluable when working with source data where words might be separated by non-standard characters (e.g., pipes, tildes, or custom symbols). The ability to precisely control what constitutes a word boundary allows for highly accurate searching even in unconventional textual formats.

Another crucial operational detail is the function's default behavior regarding case. By design, **INDEXW** conducts a case-sensitive search; consequently, a search for 'Pig' will not successfully match the word 'pig'. To ensure a robust, case-insensitive search, the recommended industry best practice involves synergistically combining **INDEXW** with other powerful SAS functions, specifically **UPCASE** or **LOWCASE**. By consistently converting both the source string and the search term to a uniform case (e.g., `INDEXW(UPCASE(phrase), UPCASE('pig'))`), developers eliminate the possibility that simple case variations will negatively impact the precision or success of the search operation. While **INDEXW** itself is an extremely efficient function, it is important to note that adding constant case conversion operations can introduce slight overhead when processing massive datasets; therefore, performance testing on representative data samples is always advisable.

Conclusion and Further Learning

The **INDEXW** function is an essential and precise tool for any SAS developer focused on text analysis and data quality. Its specialized capability to accurately locate and identify whole, standalone words by strictly honoring boundary delimiters fundamentally differentiates it from

simpler search utilities like the **INDEX** function, which only targets substrings. This distinction makes **INDEXW** superior for all analytical tasks that demand semantic accuracy, robust keyword identification, and reliable text segmentation.

By achieving mastery of the syntax, gaining a clear understanding of its optional parameters, and internalizing the stringent word-matching logic of **INDEXW**, you can dramatically improve your efficiency in cleaning, transforming, and extracting crucial, actionable insights from complex character data. The practical, step-by-step examples provided within this guide have effectively demonstrated the function's straightforward implementation and have critically highlighted the functional differences that make **INDEXW** the mandated choice for specific, high-precision text search requirements.

We strongly advocate that you continue your development journey by actively experimenting with the advanced features of **INDEXW**, particularly exploring the utility of custom delimiter options. These options allow you to skillfully adapt the function to process highly diverse and non-standard textual formats with consistent accuracy. Mastering foundational SAS functions like **INDEXW** is a critical step toward maximizing your overall efficiency and deepening your analytical capabilities within the powerful SAS environment. We encourage you to further your learning by examining other related SAS string functions and exploring the breadth of their practical applications in data preparation.

Additional Resources

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