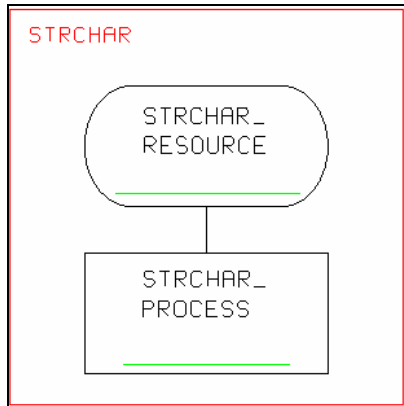


## CHARACTERS AND STRINGS

Like any other software development tool, the VSE language has its own behavior when it comes to strings and characters.

We'll start with a simple example and gradually build upon it. For starters let's build our Task module as follows:

### Example 1



### Code for task "STRCHAR"

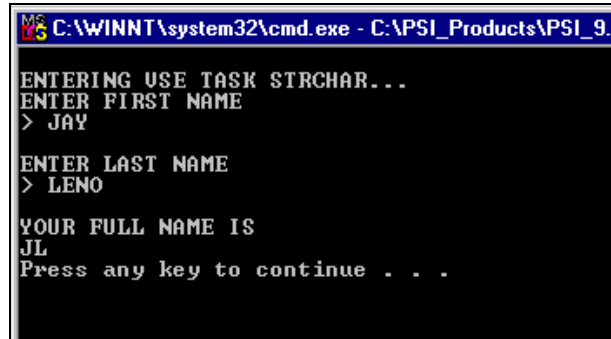
```
STRCHAR.GSS
1 CONTROL SECTION
2   TITLE, STRCHAR
3
4 MODULE SECTION
5   STRCHAR_PROCESS
6
7 END
8
```

Next we write the code for our resource module "STRCHAR\_RESOURCE" and process module "STRCHAR\_PROCESS".

```
STRCHAR_RESOURCE.RES
1 ***STRCHAR_RESOURCE
2
3 STRCHAR_RESOURCE
4   1 FIRST_NAME CHAR
5   1 LAST_NAME  CHAR
```

```
STRCHAR_PROCESS.PRO
1 ***STRCHAR_PROCESS
2
3 STRCHAR_PROCESS
4   DISPLAY 'ENTER FIRST NAME'
5   ACCEPT FIRST_NAME
6   DISPLAY 'ENTER LAST NAME'
7   ACCEPT LAST_NAME
8   DISPLAY 'YOUR FULL NAME IS'
```

On preparing the program and running it we get the following:



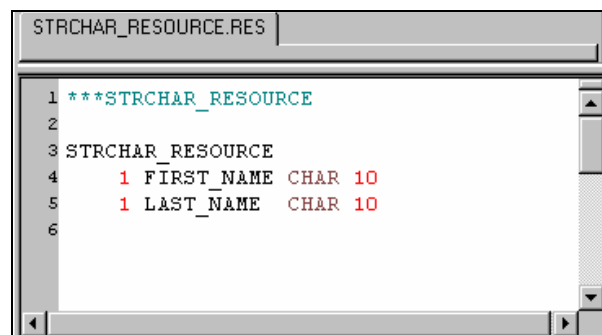
```
C:\WINNT\system32\cmd.exe - C:\PSI_Products\PSI_9.
ENTERING USE TASK STRCHAR...
ENTER FIRST NAME
> JAY
ENTER LAST NAME
> LEMO
YOUR FULL NAME IS
JL
Press any key to continue . . .
```

### Conclusions:

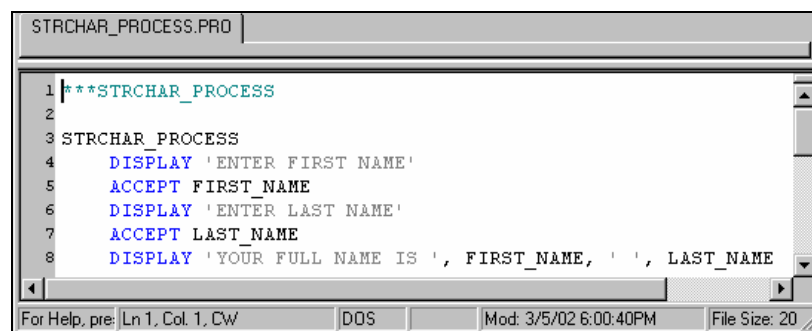
1. Clearly, the format for declaring characters [*var* CHAR] treats *var* as a single character variable. Although it seems to accept a string during input, the output clearly shows that only the first character of each of the variables is being stored.
2. In other words, the above format for declaring variables is valid for single character variables only.

### Example 2

Let's modify the code of the STRCHAR\_RESOURCE and the STRCHAR\_PROCESS to reflect the following:

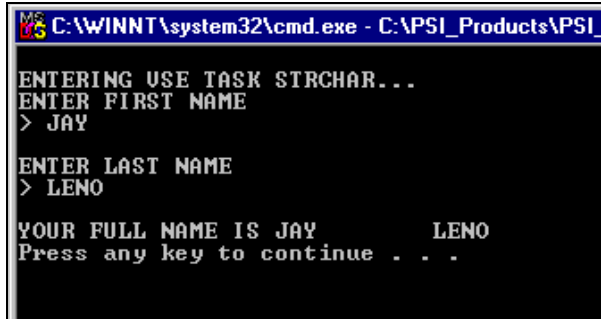


```
STRCHAR_RESOURCE.RES
1 ***STRCHAR_RESOURCE
2
3 STRCHAR_RESOURCE
4     1 FIRST_NAME CHAR 10
5     1 LAST_NAME CHAR 10
6
```



```
STRCHAR_PROCESS.PRO
1 ***STRCHAR_PROCESS
2
3 STRCHAR_PROCESS
4     DISPLAY 'ENTER FIRST NAME'
5     ACCEPT FIRST_NAME
6     DISPLAY 'ENTER LAST NAME'
7     ACCEPT LAST_NAME
8     DISPLAY 'YOUR FULL NAME IS ', FIRST_NAME, ' ', LAST_NAME
For Help, pre: Ln 1, Col 1, CW      DOS      Mod: 3/5/02 6:00:40PM      File Size: 20
```

On preparing the program and running it we get the following:



```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Products\PSI_...
ENTERING USE TASK STRCHAR...
ENTER FIRST NAME
> JAY

ENTER LAST NAME
> LENO

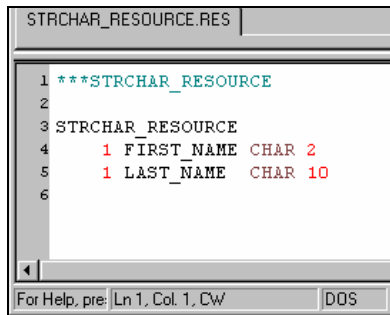
YOUR FULL NAME IS JAY          LENO
Press any key to continue . . .
```

**Conclusions:**

1. Declaring the character variables in the format [var CHAR n] gives each of the variables the capacity to store 'n' number of characters.
2. While accepting the input, the character variable of size 'n' will accept any number of characters between 1 and 'n'.
3. While printing the character variable of size 'n', it maintains a fixed field width of 'n' characters.

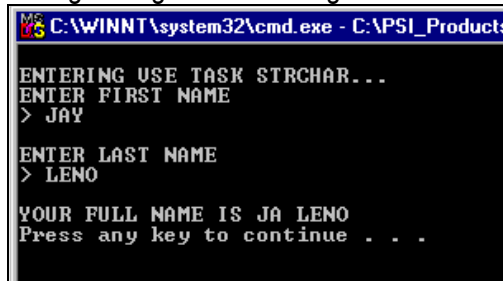
**Example 3**

In our next example, we'll modify the STRCHAR\_RESOURCE to reflect the following:



```
STRCHAR_RESOURCE.RES
1 ***STRCHAR_RESOURCE
2
3 STRCHAR_RESOURCE
4     1 FIRST_NAME CHAR 2
5     1 LAST_NAME CHAR 10
6
For Help, pre: Ln 1, Col 1, CW  DDS
```

On preparing the program and running it we get the following:



```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Products
ENTERING USE TASK STRCHAR...
ENTER FIRST NAME
> JAY

ENTER LAST NAME
> LENO

YOUR FULL NAME IS JA LENO
Press any key to continue . . .
```

**Conclusions:**

For a character variable of size 'n', if the string being stored is greater than 'n' then only the first 'n' characters of the string will be stored.

### Example 4

Defining both 'FIRST\_NAME' and 'LAST\_NAME' as character variables of size 10 each, let's rewrite the process code and try something new.

```
STRCHAR_PROCESS.PRO
1
2 ***STRCHAR_PROCESS
3
4 STRCHAR_PROCESS
5     DISPLAY 'ENTER FIRST NAME'
6     ACCEPT FIRST_NAME
7     DISPLAY 'ENTER LAST NAME'
8     ACCEPT LAST_NAME
9     IF FIRST_NAME IS EQUAL TO LAST_NAME
10        DISPLAY 'BOTH YOUR NAMES ARE SAME'
11    ELSE DISPLAY 'BOTH YOUR NAMES ARE DIFFERENT'.
```

On preparing the program and running it we get the following:

```
C:\WINNT\system32\cmd.exe - C:\PSI_Produ
ENTERING USE TASK STRCHAR...
ENTER FIRST NAME
> JAY

ENTER LAST NAME
> JAY

BOTH YOUR NAMES ARE SAME
Press any key to continue . . . _
```

```
C:\WINNT\system32\cmd.exe - C:\PSI_Products\F
ENTERING USE TASK STRCHAR...
ENTER FIRST NAME
> JAY

ENTER LAST NAME
> LENO

BOTH YOUR NAMES ARE DIFFERENT
Press any key to continue . . . _
```

### Conclusions:

Clearly, characters and strings of characters can be compared just like numbers.

### Example 5

Let's change the STRCHAR\_PROCESS code to reflect the following:

```
STRCHAR_PROCESS.PRO
1
2 ***STRCHAR_PROCESS
3
4 STRCHAR_PROCESS
5     DISPLAY 'ENTER FIRST NAME'
6     ACCEPT FIRST_NAME
7     DISPLAY 'ENTER LAST NAME'
8     ACCEPT LAST_NAME
9     IF FIRST_NAME IS LESS THAN LAST_NAME
10        DISPLAY 'FIRST NAME PRECEDES LAST NAME'.
11    IF FIRST_NAME IS GREATER THAN LAST_NAME
12        DISPLAY 'LAST NAME PRECEDES FIRST NAME'.
```

On preparing the program and running it we get the following:

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Pro
ENTERING USE TASK STCHAR...
ENTER FIRST NAME
> A

ENTER LAST NAME
> B

FIRST NAME PRECEDES LAST NAME
Press any key to continue . . .
```

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Pro
ENTERING USE TASK STCHAR...
ENTER FIRST NAME
> BUSH

ENTER LAST NAME
> JAY

FIRST NAME PRECEDES LAST NAME
Press any key to continue . . .
```

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Pro
ENTERING USE TASK STCHAR...
ENTER FIRST NAME
> B

ENTER LAST NAME
> A

LAST NAME PRECEDES FIRST NAME
Press any key to continue . . .
```

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Pro
ENTERING USE TASK STCHAR...
ENTER FIRST NAME
> JAY

ENTER LAST NAME
> JAX

LAST NAME PRECEDES FIRST NAME
Press any key to continue . . .
```

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Pro
ENTERING USE TASK STCHAR...
ENTER FIRST NAME
> a

ENTER LAST NAME
> A

LAST NAME PRECEDES FIRST NAME
Press any key to continue . . .
```

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Pro
ENTERING USE TASK STCHAR...
ENTER FIRST NAME
> Jay

ENTER LAST NAME
> jay

FIRST NAME PRECEDES LAST NAME
Press any key to continue . . .
```

### Conclusions:

1. Character variables can be sorted using normal alphabetical ordering.
2. In other words, single characters and character strings can be used in all logical comparisons just like numbers.
3. Characters and strings are sorted according to their place in the ASCII table i.e. capital letters precede small letters.
4. In other words, VSE treats single characters and strings according to their ASCII values.

### Notes

So far we have been declaring characters in the format [var CHAR n]. If 'n' is not specified the number of characters stored in 'var' is exactly one. If 'n' is specified, then 'n' number of characters can be stored in 'var'.

In our next few examples, we will use a new format for storing strings in character variables. As will be seen this format is better suited to string operations in VSE. This new format is more of an array representation of characters and hence individual characters in a string can be accessed using indexing.

## Example 6

Let's modify STRCHAR\_RESOURCE and STRCHAR\_PROCESS to reflect the following code.

```
STRCHAR_RESOURCE.RES
1 |
2 |
3 | ***STRCHAR_RESOURCE
4 |
5 | STRCHAR_RESOURCE
6 |     1 FIRST_NAME QUANTITY(10) CHAR
7 |     1 LAST_NAME  QUANTITY(10) CHAR
8 |
```

```
STRCHAR_PROCESS.PRO
1 |
2 | ***STRCHAR_PROCESS
3 |
4 | STRCHAR_PROCESS
5 |     DISPLAY 'ENTER FIRST NAME'
6 |     ACCEPT FIRST_NAME
7 |     DISPLAY 'ENTER LAST NAME'
8 |     ACCEPT LAST_NAME
9 |     DISPLAY 'YOUR FULL NAME IS ', FIRST_NAME, LAST_NAME
10 |
```

On preparing all **resources** you will see that it give us **no error**. However, on preparing all **processes** we encounter **errors**; the error in STRCHAR\_PROCESS being that the character variables FIRST\_NAME and LAST\_NAME have to be used as FIRST\_NAME (10) and LAST\_NAME (10) respectively. Let's change the code in our process to reflect the following:

```
STRCHAR_PROCESS.PRO
1 |
2 | ***STRCHAR_PROCESS
3 |
4 | STRCHAR_PROCESS
5 |     DISPLAY 'ENTER FIRST NAME'
6 |     ACCEPT FIRST_NAME(10)
7 |     DISPLAY 'ENTER LAST NAME'
8 |     ACCEPT LAST_NAME(10)
9 |     DISPLAY 'YOUR FULL NAME IS ', FIRST_NAME(10), LAST_NAME(10)
10 |
```

All our preparations should go through smoothly now. Let's run our program.

```
C:\WINNT\system32\cmd.exe - C:\PSI_Pr
ENTERING USE TASK STRCHAR...
ENTER FIRST NAME
> JAY
ENTER LAST NAME
> LENO
YOUR FULL NAME IS JL
Press any key to continue . . .
```

Surprisingly, our output looks exactly like in Example 1.

Before we make any conclusions let's modify the process code to reflect the following:

```
STRCHAR_PROCESS.PROD
1 ***STRCHAR_PROCESS
2
3 STRCHAR_PROCESS
4     DISPLAY 'ENTER FIRST NAME'
5     ACCEPT FIRST_NAME(10)
6     DISPLAY 'ENTER LAST NAME'
7     ACCEPT LAST_NAME(10)
8     DISPLAY 'YOUR FULL NAME IS ', FIRST_NAME(2), LAST_NAME(2)
```

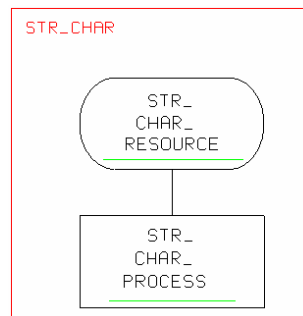
Not only does our process prepare well but it also gives us an ambiguous output.

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Pro
ENTERING USE TASK STRCHAR...
ENTER FIRST NAME
> JAY
ENTER LAST NAME
> LENO
YOUR FULL NAME IS
Press any key to continue . . .
```

Clearly, we aren't respecting VSE's rules of handling character arrays. Let's try some more examples.

### Example 7

Let's build our Task module and build the code as follows:



```
STR_CHAR.GSS
1 CONTROL SECTION
2     TITLE, STR_CHAR
3
4 MODULE SECTION
5     STR_CHAR_PROCESS
6
7 END
```

We now change our Resource and Process code to reflect the following:

```
STR_CHAR_RESOURCE.RES
1 ***STR_CHAR_RESOURCE
2 STR_CHAR_RESOURCE
3
4 ***INPUT PARAMETERS
5     1 FIRST_NAME
6     2 FIRST_NAME_CHAR QUANTITY(10) CHAR
7     1 LAST_NAME
8     2 LAST_NAME_CHAR QUANTITY (10) CHAR
```

```

STR_CHAR_PROCESS.PRO
1  ***STR_CHAR_PROCESS
2
3  STR_CHAR_PROCESS
4
5  DISPLAY 'ENTER FIRST NAME'
6  ACCEPT FIRST_NAME
7  DISPLAY 'ENTER LAST NAME'
8  ACCEPT LAST_NAME
9  DISPLAY 'YOUR FULL NAME IS ', FIRST_NAME, LAST_NAME

```

The output is as follows:

```

C:\WINNT\system32\cmd.exe - C:\PSI_Prod
ENTERING USE TASK STR_CHAR...
ENTER FIRST NAME
> JAY

ENTER LAST NAME
> LENO

YOUR FULL NAME IS JAY          LENO
Press any key to continue . . .

```

As we can see, the output is the same as in Example 2.

### Example 8

Now, look at our next example. We accept two strings as first name and last name. We then print the output as individual characters in two columns.

```

STR_CHAR_RESOURCE.RES
1  ***STR_CHAR_RESOURCE
2  STR_CHAR_RESOURCE
3
4  ***INPUT PARAMETERS
5      1 FIRST_NAME
6          2 FIRST_NAME_CHAR QUANTITY(10) CHAR
7      1 LAST_NAME
8          2 LAST_NAME_CHAR QUANTITY (10) CHAR
9
10     1 NAME_INDEX INDEX
11     1 NAME_LEN INDEX

```

```

STR_CHAR_PROCESS.PRO
1  ***STR_CHAR_PROCESS
2
3  STR_CHAR_PROCESS
4  DISPLAY 'ENTER FIRST NAME'
5  ACCEPT FIRST_NAME
6  DISPLAY 'ENTER LAST NAME'
7  ACCEPT LAST_NAME
8  DISPLAY 'YOU ENTERED ', FIRST_NAME, LAST_NAME
9  DISPLAY 'HERE IS THE ANALYSIS OF YOUR NAMES:'
10 NAME_LEN = 11
11 NAME_INDEX = 0
12 EXECUTE DISPLAY_NAMES
13     INCREMENTING NAME_INDEX FROM 0
14     UNTIL NAME_INDEX EQUALS NAME_LEN
15
16 DISPLAY_NAMES
17     DISPLAY FIRST_NAME_CHAR(NAME_INDEX), ' ',
18     LAST_NAME_CHAR(NAME_INDEX)

```

The output is as shown:

```
C:\WINNT\system32\cmd.exe - C:\PSI_Product
ENTERING USE TASK STR_CHAR...
ENTER FIRST NAME
> JENNIFER

ENTER LAST NAME
> LOPEZ

YOU ENTERED JENNIFER LOPEZ
HERE IS THE ANALYSIS OF YOUR NAMES:

J L
E O
N P
N E
I Z
F E
R
```

### Conclusions:

1. VSE allows string inputs. But it also allows us to fetch individual characters from a given input.
2. Syntactically however, we are only dealing with characters.

### Example 9

We next process a string input and output the string in the reverse order of its characters.

```
STR_CHAR_PROCESS.PRO
1 |***STR_CHAR_PROCESS
2
3 STR_CHAR_PROCESS
4   DISPLAY 'ENTER FIRST NAME'
5   ACCEPT FIRST_NAME
6   DISPLAY 'ENTER LAST NAME'
7   ACCEPT LAST_NAME
8   DISPLAY 'YOU ENTERED ', FIRST_NAME, LAST_NAME
9   DISPLAY 'HERE IS THE ANALYSIS OF YOUR NAMES:'
10  NAME_LEN = 0
11  NAME_INDEX = 0
12  EXECUTE DISPLAY_NAMES
13      DECREMENTING NAME_INDEX FROM 10
14      UNTIL NAME_INDEX EQUALS NAME_LEN
15
16 DISPLAY_NAMES
17     DISPLAY FIRST_NAME_CHAR(NAME_INDEX), ' ',
18     LAST_NAME_CHAR(NAME_INDEX)
```

Our output looks as follows:

```
C:\WINNT\system32\cmd.exe - C:\PSI_Product
ENTERING USE TASK STR_CHAR...
ENTER FIRST NAME
> JENNIFER

ENTER LAST NAME
> LOPEZ

YOU ENTERED JENNIFER LOPEZ
HERE IS THE ANALYSIS OF YOUR NAMES:

R
E
F
I Z
N E
N P
E O
J L
Press any key to continue . . .
```

## Example 10

As part of our experiment to break strings up, let's now write a program that will allow the user to enter text. The program will then tell the user the total number of characters in the text he entered.

```
STR_CHAR_RESOURCE.RES*
1 ***STR_CHAR_RESOURCE
2 STR_CHAR_RESOURCE
3
4 ***INPUT PARAMETERS
5     1 NOTES
6     2 NOTES_CHAR QUANTITY(87) CHAR
7
8     1 NO_OF_CHARS INTEGER
```

```
STR_CHAR_PROCESS.PRO
1 |***STR_CHAR_PROCESS
2
3 STR_CHAR_PROCESS
4     NO_OF_CHARS = 87 ***STANDS FOR MAX CHARS ALLOWED
5     EXECUTE MAIN_FUNCTION
6
7 MAIN_FUNCTION
8     DISPLAY 'ENTER NOTES'
9     ACCEPT  NOTES
10    EXECUTE  CALCULATE_NOTES_LENGTH
11    DISPLAY 'NUMBER OF CHARACTERS IN YOUR NOTES: ', NO_OF_CHARS
12
13 CALCULATE_NOTES_LENGTH
14    EXECUTE CHECK_SPACES
15    DECREMENTING NO_OF_CHARS FROM 87
16    UNTIL NO_OF_CHARS IS EQUAL TO ZERO
17    OR NOTES_CHAR(NO_OF_CHARS) IS NOT EQUAL TO SPACE
18
19 CHECK_SPACES
20 ***DUMMY RULE
```

```
MS-DOS C:\WINNT\system32\cmd.exe - C:\PSI_Products\PSI_9.2.1\
ENTERING USE TASK STR_CHAR...
ENTER NOTES
> Jay Leno
NUMBER OF CHARACTERS IN YOUR NOTES:      8
Press any key to continue . . .
```

### VSE BUG / ANOMALY:

When the maximum characters (NO\_OF\_CHARS) defined

1. Are 87 or less characters it shows total chars as 87 or less (program behaves as expected)
  2. Are 88 characters it shows total chars as 0 irrespective of the number of characters entered.
  3. Are 89+ characters it shows total chars as 89+ irrespective of the number of characters entered.
- I lost a great deal of time figuring out why my program doesn't work because I had my max characters initialized to 100.

## Example 11

We now improvise on our previous example to count words instead of just characters.

```
STR_CHAR_RESOURCE.RES
1 ***STR_CHAR_RESOURCE
2 STR_CHAR_RESOURCE
3
4 ***INPUT PARAMETERS
5     1 NOTES
6         2 NOTES_CHAR QUANTITY(87) CHAR
7
8     1 NO_OF_CHARS  INTEGER INITIAL_VALUE 87
9     1 NO_OF_WORDS  INTEGER INITIAL_VALUE 0
10    1 TEMP_INDEX   INTEGER INITIAL_VALUE 0
11
12    ***CHECKING IF FIRST CHAR READING IN REVERSE
13    ***IS A SPACE
14    1 IS_SPACE     INTEGER INITIAL_VALUE 0
```

```
STR_CHAR_PROCESS.PROD
1 ***STR_CHAR_PROCESS
2
3 STR_CHAR_PROCESS
4     EXECUTE MAIN_FUNCTION
5
6 MAIN_FUNCTION
7     DISPLAY 'ENTER NOTES'
8     ACCEPT  NOTES
9     EXECUTE CALCULATE_NO_OF_CHARS
10    DISPLAY 'NUMBER OF CHARACTERS IN YOUR NOTES: ', NO_OF_CHARS
11    TEMP_INDEX = NO_OF_CHARS
12    EXECUTE CALCULATE_NO_OF_WORDS
13    INCREMENT NO_OF_WORDS
14    DISPLAY 'NUMBER OF WORDS: ', NO_OF_WORDS
15
16 CALCULATE_NO_OF_WORDS
17     EXECUTE SCAN_NOTES DECREMENTING TEMP_INDEX FROM NO_OF_CHARS
18     UNTIL TEMP_INDEX IS EQUAL TO ZERO
19
20 CALCULATE_NO_OF_CHARS
21     EXECUTE CHECK_SPACES
22     DECREMENTING NO_OF_CHARS FROM 87
23     UNTIL NO_OF_CHARS IS EQUAL TO ZERO
24     OR NOTES_CHAR(NO_OF_CHARS) IS NOT EQUAL TO SPACE
25
26 SCAN_NOTES
27     IF NOTES_CHAR(TEMP_INDEX) IS EQUAL TO SPACE
28     EXECUTE UPDATE_WORDS.
29
30 UPDATE_WORDS
31     INCREMENT NO_OF_WORDS BY 1
32
33 CHECK_SPACES
34 ***DUMMY RULE
```

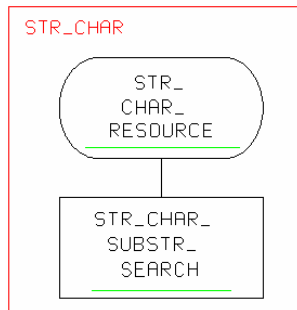
```
C:\WINNT\system32\cmd.exe - C:\PSI_Products\PSI_9.2.1\SS
ENTERING USE TASK STR_CHAR...
ENTER NOTES
> THREE WORDS HERE

NUMBER OF CHARACTERS IN YOUR NOTES:      16
NUMBER OF WORDS:                          3
Press any key to continue . . . _
```

**Note:** The above is a basic program and does not work well when the sentence is preceded by spaces since we don't account for spaces at the beginning of the string.

## Example 12

Our last example allows the user to enter notes. It also allows finding a string within those notes. The output tells the user whether the string to be found exists or not. This program is based on substring searches.



```
STR_CHAR.GSS
1 CONTROL SECTION
2     TITLE, STR_CHAR
3
4 MODULE SECTION
5     STR_CHAR_SUBSTR_SEARCH
6
7
8 END
```

### STR\_CHAR\_RESOURCE.RES

```
WORD ***WORD TO BE FOUND
1 WORD_CHAR QUANTITY(87) CHAR
NOTES ***THE NOTES THAT USER WILL ENTER
1 NOTES_CHAR QUANTITY(87) CHAR

SEARCH_RESPONSE STATUS FOUND, NOT_FOUND
COMPARE_RESPONSE STATUS UNDER_TEST, LESSTHAN, GREATERTHAN, MATCHED

WORD_LEN INDEX
NOTES_LEN INDEX
WORD_IDX1 INDEX
WORD_IDX2 INDEX
NOTES_IDX1 INDEX
NOTES_IDX2 INDEX
```

### CODE: STR\_CHAR\_SUBSTR\_SEARCH

```
SEARCH_SUBSTRING
EXECUTE ACCEPT_STRINGS
EXECUTE CALCULATE_WORD_LENGTH
EXECUTE CALCULATE_NOTES_LENGTH
EXECUTE SUBSTRING_TEST
IF SEARCH_RESPONSE IS FOUND DISPLAY 'FOUND -> ',WORD
ELSE DISPLAY 'NOT FOUND -> ',WORD.

ACCEPT_STRINGS
DISPLAY 'ENTER YOUR NOTES'
ACCEPT NOTES
DISPLAY 'ENTER THE WORD YOU WANT TO FIND'
ACCEPT WORD
DISPLAY '-----'
DISPLAY 'YOUR NOTES ARE -> ', NOTES
DISPLAY 'AND YOU WANT TO FIND -> ', WORD
DISPLAY '-----'

CALCULATE_WORD_LENGTH
EXECUTE SCAN_WORD
DECREMENTING WORD_LEN FROM 64
UNTIL WORD_LEN IS EQUAL TO ZERO
OR WORD_CHAR(WORD_LEN) IS NOT EQUAL TO SPACE

SCAN_WORD
***DUMMY RULE, JUST FOR LOOP
```

```

CALCULATE_NOTES_LENGTH
EXECUTE SCAN_NOTES
    DECREMENTING NOTES_LEN FROM 64
    UNTIL NOTES_LEN IS EQUAL TO ZERO
        OR NOTES_CHAR(NOTES_LEN) IS NOT EQUAL TO SPACE

SCAN_NOTES
***DUMMY RULE, JUST FOR LOOP

SUBSTRING_TEST
SET SEARCH_RESPONSE TO NOT_FOUND

    IF WORD_LEN IS GREATER THAN NOTES_LEN
        EXIT THIS RULE .
    IF NOTES_LEN IS LESS THAN 1
        EXIT THIS RULE .

    IF WORD_LEN IS LESS THAN 1
***CONSIDER AN EMPTY STRING AS A SUBSTRING OF THE OTHER
    SET SEARCH_RESPONSE TO FOUND
    EXIT THIS RULE .

EXECUTE SEARCH_WORD_OVER_NOTES
    INCREMENTING NOTES_IDX1
    UNTIL NOTES_IDX1 IS GREATER THAN NOTES_LEN
        OR SEARCH_RESPONSE IS FOUND

SEARCH_WORD_OVER_NOTES
WORD_IDX1 = 1
EXECUTE COMPARE_EACH_CHAR
    INCREMENTING NOTES_IDX2 FROM NOTES_IDX1
    UNTIL NOTES_IDX2 IS GREATER THAN NOTES_LEN
        OR WORD_IDX1 IS GREATER THAN WORD_LEN
        OR WORD_CHAR(WORD_IDX1) IS NOT EQUAL TO
        NOTES_CHAR(NOTES_IDX2)

    IF WORD_IDX1 IS GREATER THAN WORD_LEN
***WORD IS FOUND IN NOTES
    SET SEARCH_RESPONSE TO FOUND .

COMPARE_EACH_CHAR
    INCREMENT WORD_IDX1

```

<pre> C:\WINNT\system32\cmd.exe - C:\PSI_Products\PSI_9.2.1\SPEC_L ENTERING USE TASK STR_CHAR... ENTER YOUR NOTES &gt; lets see if we can find the word.  ENTER THE WORD YOU WANT TO FIND &gt; see  ----- YOUR NOTES ARE -&gt; lets see if we can find the word. AND YOU WANT TO FIND -&gt; see  ----- FOUND -&gt; see Press any key to continue . . . _ </pre>	<pre> C:\WINNT\system32\cmd.exe - C:\PSI_Products\PSI_9.2.1\SPEC_LINK_AN ENTERING USE TASK STR_CHAR... ENTER YOUR NOTES &gt; Let's see if we can find the word.  ENTER THE WORD YOU WANT TO FIND &gt; JAY  ----- YOUR NOTES ARE -&gt; Let's see if we can find the word. AND YOU WANT TO FIND -&gt; JAY  ----- NOT FOUND -&gt; JAY Press any key to continue . . . _ </pre>
---	---