

ARB05

COLLABORATORS

	<i>TITLE :</i> ARB05		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ARB05	1
1.1	ARexx For Beginners - Article 5 - Putting Info Into Window	1
1.2	Article 5 - Putting Info Into Window - General Comments	1
1.3	Article 5 - Putting Info Into Window - The SAY & ECHO Commands	2
1.4	Article 5 - Putting Info Into Window - String Delimiters	3
1.5	Article 5 - Putting Info Into Window - Inserting Inverted Commas & Apostrophes	4
1.6	Article 5 - Putting Info Into Window - Which Delimiter To Use	5
1.7	Article 5 - Putting Info Into Window - Using SAY Without String Delimiters	5

Chapter 1

ARB05

1.1 ARexx For Beginners - Article 5 - Putting Info Into Window

AREXX FOR BEGINNERS

ARTICLE 5 - PUTTING INFORMATION INTO THE WINDOW

BY FRANK BUNTON

COPYRIGHT © FRANK P. BUNTON 1995-1998

General Comments

The SAY Command

String Delimiters

Inserting Inverted Commas & Apostrophes

Which Delimiter To Use

Using SAY Without String Delimiters

=== End of Text ===

1.2 Article 5 - Putting Info Into Window - General Comments

GENERAL COMMENTS

The first thing you should learn in any programming language is how to put information into the display window. To do this you use the "SAY" or "ECHO" instructions. They are interchangeable. However, to avoid confusion with the AmigaDOS command "ECHO" (which can be called from an ARexx script) I will only use "SAY".

Do NOT confuse "SAY" with the AmigaDOS "SAY" command which produces artificial speech. (Note - AmigaDOS "SAY" is only available in Workbench2.0 and earlier. It was dropped from Workbench2.1 and later.)

=== End of Text ===

1.3 Article 5 - Putting Info Into Window - The SAY & ECHO Commands

THE SAY & ECHO COMMANDS

Please note that the ECHO command is identical to SAY and all the following applies to ECHO as well as to SAY

SAY can be used in one of these forms:-

```
SAY
SAY Expression
```

SAY on its own will produce a blank line.

SAY followed by an expression results in the expression being displayed in the output window.

The "Expression" can be one of these things:-

- a string - if so enclose it in delimiters (quotes).
- a number.
- a mathematical expression to be evaluated.
- a symbol (see Article 7 - Symbols Introduction)

(Note - there are other things but that is enough for the beginner to be going on with!)

Examples are shown below. These examples do not include SAYing a symbol as I will leave that until we start to deal with symbols in Article 7.

In these examples, the result of the instruction clause (i.e. what appears in the output window) is shown below it AFTER the "-->" symbol).

(Don't forget what I have previously said about using example programs - this is the last reminder you will get!)

```
/* Example5-1 */
/* This illustrates SAYing a string */

SAY 'I am a String'
--> I am a String
```

Example5-1 SAYs a string and everything within the quotes appears in the output window (without the quotes, of course).

```
/* Example5-2 */
/* This illustrates SAYing a number */

SAY 45.3
--> 45.3
```

Example5-2 SAYs the number itself.

```
/* Example5-3 */
/* This illustrates SAYing a mathematical expression to be evaluated */
```

```
SAY 3 + 4 * 5
--> 23
```

Example5-3 has "numbers" and operators. Operators are items such as the addition sign (+) and the multiplication sign (*). I will discuss these in Article 9. The numbers and operators are not displayed in the output window as such but the expression following the SAY is calculated and the result of the calculation is displayed.

```
/* Example5-4 */
/* This illustrates SAYing a mathematical expression */
/* which is within a string as well as one not in a string */

SAY '3 + 4 * 5 =' 3 + 4 * 5
--> 3 + 4 * 5 = 23
```

In Example5-4, the part enclosed in quotes (') is a string and so is NOT evaluated but displayed exactly as is. The part after the closing quote is NOT a string so it IS evaluated.

=== End of Text ===

1.4 Article 5 - Putting Info Into Window - String Delimiters

STRING DELIMITERS

When a string is used it must be enclosed in string delimiters. (See using SAY without delimiters.)

A string delimiter is a character at each end of the string to define its start and finish, i.e. its limits. The delimiter character itself does not form part of the string.

There are two characters that you can use as delimiters. They are:-

- the inverted comma (") which is also known as double quote
- the apostrophe (') which is also known as the closing single quote

"Closing Single Quote" is a bit of a misnomer in ARexx terms as, if you use it ('), you must use it at the start and end of the quote! You can NOT use any other character such as the opening single quote (`).

You CAN use the two different delimiters in the same program but you must NOT mix the two string delimiters in the same string. For example:-

```
/* Example5-5 */
/* This program has an error and so will not run */

SAY "I am using inverted commas" /* this is OK */
SAY 'I am using apostrophes' /* this is OK */
SAY "This is not allowed" /* this is NOT OK */
```

Actually, this program will not run at all. An error message will be given

indicating:-

```
+++ Error 5 in line 6: Unmatched quote
Command returned 10/5: Unmatched quote
```

I will discuss the meaning of this sort of error message in Article 46.

ARexx searches the whole script before executing any instructions and looks for certain types of errors such as this one. It reports the first error it finds (if any). If there are more errors, then these will not be reported till the first one has been corrected and the program run again. Debugging and error tracing are discussed in Articles 46 to 49 inclusive.

=== End of Text ===

1.5 Article 5 - Putting Info Into Window - Inserting Inverted Commas & Apostrophes

INSERTING INVERTED COMMAS OR APOSTROPHES INTO THE TEXT

It may be that you want to insert an apostrophe or inverted comma into the output of the SAY instruction. There are two ways that you can do this. Firstly, you could use inverted commas when you want to insert an apostrophe or vice versa. Secondly, you could use two of the delimiter characters together to indicate that this is not the limit of the string but that a single instance of the delimiter character is to be inserted into the string.

To illustrate these points, try out these programs:-

```
/* Example5-6 */

SAY "He said '"I hate IBM Computers'" then left"
SAY 'He said "I hate IBM Computers" then left'
```

Both of the lines in this example produce the same output:-

```
--> He said "I hate IBM Computers" then left.
```

```
/* Example5-7 */

SAY "My sister's pet hate is IBM Computers"
SAY 'My sister''s pet hate is IBM Computers'
```

Both of the lines in this example produce the same output:-

```
--> My sister's pet hate is IBM Computers
```

=== End of Text ===

1.6 Article 5 - Putting Info Into Window - Which Delimiter To Use

WHICH DELIMITER TO USE?

I would suggest that, in order to save confusion, you pick the delimiter character that you are most happy with then stick to it, using the double delimiter if it is to be part of the text.

At first I chose inverted commas as I have used them extensively in AmigaDOS scripts and in basic programming and it became a habit. However, I soon realised that, when you run a short one line program from a Shell/CLI window, as in:-

```
> RX "SAY 'A string'"
```

you must use inverted commas (and not single quotes) to enclose the whole program line that comes after the RX. To then try to use inverted commas for the SAY instructions becomes totally confusing and extremely prone to error!!

In my opinion, therefore, it is best to always use the single quote character for the SAY commands because then you will not incorrectly use the inverted comma from force of habit when using RX in this manner. However, you can make up your own mind!

Another good reason to use ' is that to use " you must also use the shift key. You can therefore speed up your typing a bit by using '.

=== End of Text ===

1.7 Article 5 - Putting Info Into Window - Using SAY Without String Delimiters

USING SAY WITHOUT STRING DELIMITERS

I said above that, if you are SAYing a string , it should always be included in string delimiters. If you omit the delimiters, then ARexx considers it to be a symbol, not a string. The result will depend on whether the the symbol has already been assigned a value. I will discuss this further in Article 7 - "An Introduction to Symbols".

What I will say at this stage, however, is that, if you inadvertently leave out the string delimiters from around a string, you will not, in all likelihood, get the result that you expected. For example:-

```
/* Example5-8 */
```

```
SAY Number  
Number = 3  
SAY Number
```

```
--> NUMBER  
3
```


The result of the first SAY is the output of the name of the symbol "Number" converted to upper case. ARexx converts everything that is not within string delimiters to upper case.

The result of the second SAY is the value of the symbol. See Article 7 for explanations.

The moral is, always enclose your strings in string delimiters!

If you rewrote Example5-8 as follows:-

```
/* Example5-9 */
```

```
SAY 'Number'
```

```
Number = 3
```

```
SAY 'Number'
```

then the output would be:-

```
Number
```

```
Number
```

```
=== End of Text ===
```
