Filling String Arrays
-
Alyce
Alyce's Restaurant

String Array Notation

String array names end with the **$** character. Numeric array names do not have an appended character.

StringArray$()
NumericArray()

DIM String Arrays

Arrays with 11 elements, numbered 0-10 do not need to be dimensioned before use. Arrays with more elements and double-dimensioned arrays must be dimensioned.

```
DIM color$(20)
```

Assign a Value

Assign values to array indices in the same way you assign values to variables, with the **=** character. The index number appears in the parenthesis. To assign the string "Potato" to index 1 of the array named **array$()**:

```
array$(1) = "Potato"
```

Filling Arrays

You may fill an array with string literals, string variables, or string expressions.

```
array$(1)="Green"
array$(2)=upper$("red")+" "+upper$(blue)
color$="orange"
```

```
array$(3)=color$
```

Filling Arrays from Files

Arrays may be filled by inputting to an array from an open file. In the following demo a file is opened for output and filled with five color names, separated by commas. The file is closed, then opened again for input. The **input** statement reads items deliminated by commas or CRLFs. Each item in the file is assigned to an array index in a loop.

```
open "colors.txt" for output as #f
print #f, "red, green, blue, yellow, brown"
close #f

open "colors.txt" for input as #f
for i = 1 to 5
    input #f, col$(i)
next
close #f

for i = 1 to 5
    print col$(i)
next
```

Filling Arrays with WORD$()

An earlier example showed how to fill each array index individually. That requires a lot of typing and many lines of code for larger arrays. It is faster and easier to use a long string to hold the data and use the **WORD$()** function in a loop to fill the array, like so:

```
terms$ = "red green blue yellow brown"
for i = 1 to 5
    col$(i) = word$(terms$,i)
next

for i = 1 to 5
    print col$(i)
next
```

If the array elements contain more than a single word, use the optional delimiter with the **WORD$()** function. In the following code, a comma is used to separate items in a string, and then used as the delimiter in the **WORD$()** function. This allows each array element to contain a string of multiple words.

```
terms$ =
"bright red,light green,pale blue,sunshine yellow,muddy brown"
for i = 1 to 5
    col$(i) = word$(terms$,i, ",")
next

for i = 1 to 5
    print col$(i)
next
```

DATA and Arrays

The information to be loaded into an array can be contained in **DATA** statements. It is then read into a variable with the **READ** command. The value of the variable is then assigned to the array index. This can also be done in a loop in a similar fashion to reading data from a file or from a large string.

```
for i = 1 to 10
    read val$
    col$(i) = val$
next

for i = 1 to 10
    print col$(i)
next

data "bright red","light green","pale blue",
"sunshine yellow","muddy brown"
data "pewter","terra cotta","deep purple","gray","tangerine"
```

Notice that the **DATA** is **READ** into a variable. **READ** will not allow assignment directly to an array.

THE FOLLOWING CODE WILL HALT WITH A SYNTAX ERROR WHEN YOU TRY TO RUN IT

```
for i = 1 to 10
    'SYNTAX ERROR:
    read col$(i)
next

for i = 1 to 10
    print col$(i)
```

```
next

data "bright red","light green","pale blue",
"sunshine yellow","muddy brown"
data "pewter","terra cotta","deep purple","gray","tangerine"
```