A(0)	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)

Similarly, if you say

```
DIM B(2,5)
```

XYBASIC will create space for a two dimensional array named B with 3 * 6 = 18 elements. You can think of B as a rectangular collection of variables, with the first subscript specifying the row and the second specifying the column of a variable:

B(0,0)	B(0,1)	B(0,2)	B(0,3)	B(0,4)	B(0,5)
B(1,0)	B(1,1)	B(1,2)	B(1,3)	B(1,4)	B(1,5)
B(2,0)	B(2,1)	B(2,2)	B(2,3)	B(2,4)	B(2,5)

XYBASIC lets you define arrays with any number of subscripts, lets you define each dimension with any numeric formula, and lets you use a single DIM command to DIMension more than one array. In Extended XYBASIC you can also define arrays of any type -- floating point, integer, or string. For example,

```
DIM S$ (I), B (2, I * J), X% (2, 2, 2, 2)
```

tells Extended XYBASIC to set aside space for a one dimensional string array S$ with I + 1 elements, a two dimensional floating point array B with 3 * (I * J + 1) elements, and a four dimensional integer array X% with 3 * 3 * 3 * 3 = 81 elements. More information about string arrays is given in Section 4.

The following example demonstrates how array elements may be manipulated.

```
NEW
OK
10 DIM A(10)
20 A(0) = 1
30 A(1) = 1
40 FOR I = 2 TO 10
50 A(I) = A(I-1) + A(I-2)
60 NEXT I
70 FOR I = 0 TO 10
80 PRINT A(I);
90 NEXT I
RUN
 1  1  2  3  5  8  13  21  34  55  89
OK
```

Here line 10 is a DIMension statement, creating an array named A with 11 elements. After each element is given an initial value, the FOR loop starting at line 70 prints the values.