Employing variable arrays

An array is a variable that can store multiple items of data – unlike a regular variable, which can only store one piece of data. The pieces of data are stored sequentially in array "elements" that are numbered, starting at zero. So the first value is stored in element zero, the second value is stored in element one, and so on.

Array numbering starts at zero – so the final element in an array of six elements is number five, not number six.

An array is declared in the same way as other variables but additionally the size of the array must also be specified in the declaration, in square brackets following the array name. For example, the syntax to declare an array named "nums" to store six integer numbers looks like this:

```
int nums[6] ;
```

Optionally an array can be initialized when it is declared by assigning values to each element as a comma-separated list enclosed by curly brackets (braces). For example:

```
int nums[6] = { 0, 1, 2, 3, 4, 5 } ;
```

An individual element can be referenced using the array name followed by square brackets containing the element number. This means that **nums[1]** references the second element in the example above – not the first element, as element numbering starts at zero.

Arrays can be created for any C++ data type, but each element may only contain data of the same data type. An array of characters can be used to store a string of text if the final element contains the special **\0** null character. For example:

```
char name[5] = { 'm', 'i', 'k', 'e', '\0' } ;
```

The entire string to be referenced just by the array name. This is the principle means of working with strings in the C language but the C++ string class, introduced in chapter four, is far simpler.

Collectively the elements of an array are known as an "index". Arrays can have more than one index – to represent multiple dimensions, rather than the single dimension of a regular array. Multi-dimensional arrays of three indices and more are uncommon, but two-dimensional arrays are useful to store grid-based information, such as coordinates. For example:

```
int coords[2] [3] = { { 1, 2, 3 } , { 4, 5, 6 } } ;
```

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6