Using new to Create Dynamic Arrays

If all a program needs is a single value, you might as well declare a simple variable, for that is simpler, if less impressive, than using `new` and a pointer to manage a single small data object. More typically, you use `new` with larger chunks of data, such as arrays, strings, and structures. This is where `new` is useful. Suppose, for example, you're writing a program that might or might not need an array, depending on information given to the program while it is running. If you create an array by declaring it, the space is allocated when the program is compiled. Whether or not the program finally uses the array, the array is there, using up memory. Allocating the array during compile time is called static binding, meaning the array is built in to the program at compilation time. But with `new`, you can create an array during runtime if you need it and skip creating the array if you don't need it. Or, you can select an array size after the program is running. This is called dynamic binding, meaning that the array is created while the program is running. Such an array is called a dynamic array. With static binding, you must specify the array size when you write the program. With dynamic binding, the program can decide upon an array size while the program runs.

For now, we'll look at two basic matters concerning dynamic arrays: how to use C++'s `new` operator to create an array and how to use a pointer to access array elements.

Creating a Dynamic Array with new

It's easy to create a dynamic array in C++; you tell `new` the type of array element and number of elements you want. The syntax requires that you follow the type name with the number of elements in brackets. For example, if you need an array of ten ints, do this:

```
int * psome = new int [10]; // get a block of 10 ints
```

The `new` operator returns the address of the first element of the block. In this example, that value is assigned to the pointer `psome`. You should balance the call to `new` with a call to `delete` when the program is finished with using that block of memory.

When you use `new` to create an array, you should use an alternative form of `delete` that indicates that you are freeing an array:

```
delete [] psome; // free a dynamic array
```