APPENDIX 6

Overloading the Array Index Square Brackets

You can overload the square brackets, `[]`, for a class so that they can be used with objects of the class. If you want to use `[]` in an expression on the left-hand side of an assignment operator, then the operator must be defined to return a reference, which is indicated by adding & to the returned type. (This has some similarity to what we discussed for overloading the I/O operators << and >>.) When overloading `[]`, the operator `[]` *must* be a member function; the overloaded `[]` *cannot* be a friend operator. (In this regard, `[]` is overloaded in a way similar to the way in which the assignment operator = is overloaded; overloading = is discussed in the section of Chapter 11 entitled "Overloading the Assignment Operator.")

For example, the following defines a class called `Pair` whose objects behave like arrays of characters with the two indexes 1 and 2 (*not* 0 and 1):

```
class Pair
{
public:
    Pair();
    Pair(char first_value, char second_value);
    char& operator[](int index);
private:
    char first;
    char second;
};
```

The definition of the member function `[]` can be as follows:

```
char& Pair::operator[](int index)
{
    if (index == 1)
        return first;
    else if (index == 2)
        return second;
    else
    {
        cout << "Illegal index value.\n";
        exit(1);
    }
}
```

Objects are declared and used as follows:

```
Pair a;
a[1] = 'A';
a[2] = 'B';
cout << a[1] << a[2] << endl;
```

Note that in `a[1]`, `a` is the calling object and `1` is the argument to the member function `[]`.