return 0;
}

Compatibility Note

cout is a smart object, but some versions are smarter than others. Thus, some implementations might fail to recognize pointer types. In that case, you have to type cast the address to a recognizable type, such as unsigned int. The appropriate type cast depends on the memory model.
The default DOS memory model uses a 2-byte address, hence unsigned int is the proper cast. Some DOS memory models, however, use a 4-byte address, which requires a cast to unsigned long.

Here is the output on one system:

donuts value = 6 and donuts address = 0x0065fd40
Cups value = 4.5 and cups address = 0x0065fd44

When it displays addresses, cout uses hexadecimal notation because that is the usual notation used to describe memory. Our implementation stores donuts at a lower memory location than cups. The difference between the two addresses is 0x0065fd44 - 0x0065fd40, or 4. This makes sense, for donuts is type int, which uses four bytes. Different systems, of course, will give different values for the address. Also, some may store cups first, then donuts, giving a difference of 8 bytes, since cups is double.

Using ordinary variables, then, treats the value as a named quantity and the location as a derived quantity. Now look at the pointer strategy, one that is essential to the C++ programming philosophy of memory management. (See the note on Pointers and the C++ Philosophy.)

Pointers and the C++ Philosophy

Object-oriented programming differs from traditional procedural programming in that OOP emphasizes making decisions during runtime instead of during compile time. Runtime means while a program is running, and compile