**第九章**

C++鼓励程序员在开发程序时使用多个文件。一种有效的组织策略是，使用头文件来定义用户类型，为操纵用户类型的函数提供函数原型；并将函数定义放在一个独立的源代码文件中。头文件和源代码文件起定义和实现了用户定义的类型及其使用方式。最后，将main( )和其他使用这些函数的函数放在第三个文件中。

C++的存储方案决定了变量保留在内存中的时间（储存持续性）以及程序的哪一部分可以访问它（作用域和链接性）。自动变量是在代码块（如函数体或函数体中的代码块）中定义的变量，仅当程序执行到包含定义的代码块时，它们才存在，并且可见。自动变量可以通过使用存储类型说明符register或根本不使用说明符来声明，没有使用说明符时，变量将默认为自动的。register说明符提示编译器，该变量的使用频率很高，但 C++11摒弃了这种用法。

静态变量在整个程序执行期间都存在。对于在函数外面定义的变量，其所属文件中位于该变量的定义后面的所有函数都可以使用它（文件作用域），并可在程序的其他文件中使用（外部链接性）。另一个文件要使用这种变量，必须使用 extern关键字来声明它。对于文件间共享的变量，应在一个文件中包含其定义声明（无需使用 extern但如果同时进行初始化也可使用它）并在其他文件中包含引用声明（使用 extern且不初始化）。在函数的外面使用关键字static定义的变量的作用域为整个文件，但是不能用于其他文件（内部链接性）。在代码块中使用关键字static定义的变量被限制在该代码块内（局部作用域、无链接性），但在整个程序执行期间，它都一直存在并且保持原值。

在默认情况下，C++函数的链接性为外部，因此可在文件间共享；但使用关键字static限定的函数的链接性为内部的，被限制在定义它的文件中。

动态内存分配和释放是使用new和delete进行的，它使用自由存储区或堆来存储数据。调用new占用内存，而调用delete释放内存。程序使用指针来跟踪这些内存单元。

名称空间允许定义一个可在其中声明标识符的命名区域。这样做的目的是减少名称冲突，尤其当程序非常大，并使用多个厂商的代码时。可以通过使用作用域解析运算符、using声明或using编译指令，来使名称空间中的标识符可用。

**第十章**

面向对象编程强调的是程序如何表示数据。使用OOP方法解决编程问题的第一步是根据它与程序之间的接口来描述数据，从而指定如何使用数据。然后，设计一个类来实现该接口。一般来说，私有数据成员存储信息，公有成员函数(又称为方法)提供访问数据的唯一途径。类将数据和方法组合成一个单元，其私有性实现数据隐藏。

通常，将类声明分成两部分组成，这两部分通常保存在不同的文件中。类声明（包括由函数原型表示的方法）应放到头文件中。定义成员函数的源代码放在方法文件中。这样便将接口描述与实现细节分开了从理论上说，只需知道公有接口就可以使用类。当然，可以查看实现方法（除非只提供了编译形式），但程序不应依赖于其实现细节，如知道某个值被存储为int。只要程序和类只通过定义接口的方法进行通信，程序员就可以随意地对任何部分做独立的改进，而不必担心这样做会导致意外的不良影响。

类是用户定义的类型，对象是类的实例。这意味着对象是这种类型的变量，例如由new按类描述分配的内存。C++试图让用户定义的类型尽可能与标准类型类似，因此可以声明对象、指向对象的指针和对象数组。可以按值传递对象、将对象作为函数返回值、将一个对象赋给同类型的另一个对象。如果提供了构造函数，则在创建对象时，可以初始化对象。如果提供了析构函数方法，则在对象消亡后，程序将执行该函数。

每个对象都存储自己的数据，而共享类方法。如果mr\_object是对象名，try\_me( )是成员函数，则可以使用成员运算符句点调用成员函数：mr\_object.try\_me( )。在OOP中这种函数调用被称为将try\_me消息发送给 mr\_object对象。在try\_me( )方法中引用类数据成员时，将使用mr\_object对象相应的数据成员。同样，函数调用i\_object.try\_me( )将访问i\_object对象的数据成员。

如果希望成员函数对多个对象进行操作，可以将额外的对象作为参数传递给它。如果方法需要显式地引用调用它的对象，则可以使用this指针。由于this指针被设置为调用对象的地址，因此\*this是该对象的别名。

类很适合用于描述 ADT。公有成员函数接口提供了ADT描述的服务，类的私有部分和类方法的代码提供了实现，这些实现对类的客户隐藏。