The class is the key element for object-oriented programming in C++. Classes enable a programmer to create new types that have all the features of integral types. In fact, by applying the full features of classes, user-defined types can be designed to be as powerful, well-behaved and ‘concrete’ as integral types such as int.

The class is analogous to the structure, in that a class is a data encapsulation mechanism, but with the additional support of functions specific to the class which operate on the class’s data. Both class data and member functions can have associated access rights which allow the designer of a class to either allow or restrict access to specific data members to a non-class member or an object of the class. Thus, a class can be designed with an interface to non-class members which encapsulates the functionality of the class and at the same time hides the inner details of the class.

A class defines a mould from which objects or instances can be created, just as Jean is a particular instance of the class Person. An object is a single, unique entity that comprises both data and member functions which operate on their data.

Throughout this chapter, emphasis is placed on the development of a single class called Point which characterises a point in three-dimensions. As the chapter progresses we shall steadily see the functionality of class Point grow, so that when the chapter concludes Point will be a simple yet powerful class which reflects the majority of features that are characteristic of all C++ classes.

It is important to develop a good understanding of the class because it is the cornerstone of object-oriented programming in C++ and will play an increasing role in subsequent chapters.

9.1 A Point class

This chapter introduces the C++ class, which is used in the creation of new user-defined types. Classes are introduced with the help of a Point class which is an excellent class to illustrate the key features of classes and is essential for graphics programming in C++.

Point-style classes are widely used in computer graphics. For example, in programming for Windows use is frequently made of the POINT structure defined in the WINDOWS.H header file:
The \texttt{typedef} \texttt{POINT} is extensively used to model a point on a two-dimensional display screen. Note that \texttt{tagPOINT}'s data members are of type integer.

The Borland C++ (version 5.x) compiler supports an ObjectWindows library \texttt{class} \texttt{TPoint}. The \texttt{class} \texttt{TPoint} inherits the \texttt{tagPOINT} \texttt{x} and \texttt{y} data points and provides a sufficient number of overloaded operators to enable a natural use of \texttt{TPoint} objects:

Both \texttt{tagPOINT} and \texttt{TPoint} \texttt{x} and \texttt{y} data members are \texttt{public}. The distinction between \texttt{public} and \texttt{private} will be discussed later.

In this chapter we will be interested in a more accurate representation of a Point for general three-dimensional geometric modelling. Therefore, there are three data members for our \texttt{Point} \texttt{class}, each of type \texttt{double}. Type \texttt{double} can at first appear rather memory-expensive when compared with type \texttt{float}, but type \texttt{double} is necessary for the required level of accuracy. When we examine the C++ \texttt{template} we will observe that a generic \texttt{Point} \texttt{class} can be declared with data members whose types are determined when an object of the \texttt{Point} \texttt{class} is defined.

Figure 9.1 illustrates an arbitrary point, \( p (p_x, p_y, p_z) \), in a three-dimensional space. The position of point \( p \) is given relative to a reference point or origin \( O \). The origin \( O \) can have any assigned position, but it is generally assumed that \( O \) has coordinates \( (0, 0, 0) \). Point \( O \) is the origin of a three-dimensional coordinate system that consists of three mutually perpendicular straight lines or axes. Such a rectangular coordinate system is referred to as a \textit{Cartesian} coordinate system.

Let's begin our examination of the \texttt{Point} \texttt{class} by copying the \texttt{Point} structure presented in the program POINT.CPP in Chapter 8 and simply replacing the keyword \texttt{struct} in the \texttt{Point} structure declaration by the keyword \texttt{class}:

```cpp
// class_1.cpp
// introduces the class
#include <iostream>  // C++ I/O

class Point
{
  double x, y, z;
```