Preface to the Third Edition

Object-Oriented Programming (OOP) has become the preferred programming approach by the software industries, as it offers a powerful way to cope with the complexity of real-world problems. Among the OOP languages available today, C++ is by far the most widely used language.

Since its creation by Bjarne Stroustrup in early 1980s, C++ has undergone many changes and improvements. The language was standardized in 1998 by the American National Standards Institute (ANSI) and the International Standards Organization (ISO) by incorporating not only the new features but also the changes suggested by the user groups. This book has been thoroughly revised and this edition confirms to the specifications of ANSI/ISO standards. Besides confirming to the standards, many smaller changes and additions to strengthen the existing topics as well as corrections to typographical errors and certain inaccuracies in the text have been incorporated.

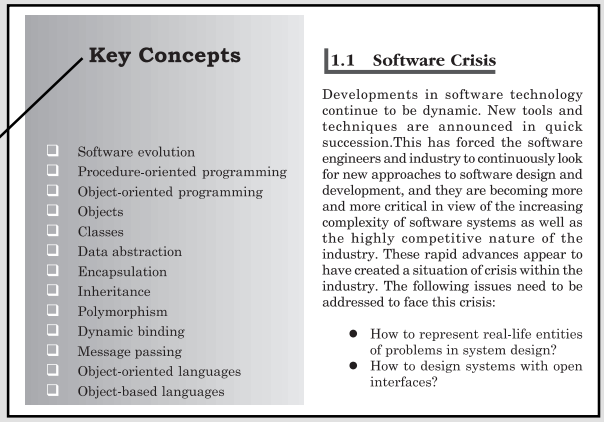
This book is for the programmers who wish to know all about C++ language and object-oriented programming. It explains in a simple and easy-to-understand style the what, why and how of object-oriented programming with C++. The book assumes that the reader is already familiar with C language, although he or she need not be an expert programmer.

The book provides numerous examples, illustrations and complete programs. The sample programs are meant to be both simple and educational. Wherever necessary, pictorial descriptions of concepts are included to improve clarity and facilitate better understanding. The book also presents the concept of object-oriented approach and discusses briefly the important elements of object-oriented analysis and design of systems.

Key Pedagogical Features

Key Concepts

Key concepts provide a quick look into the concepts that will be discussed in the chapter. These are followed by an Introduction that introduces the topics to be covered in that chapter and also relates them to those already learned.



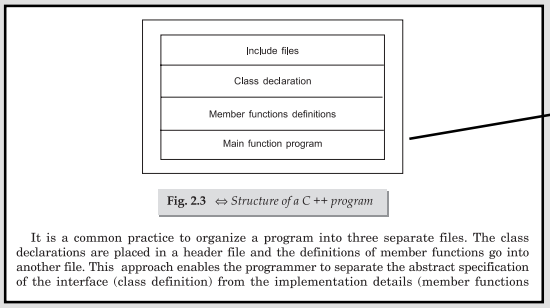


Fig. 2.3 ⇔ *Structure of a C ++ program*

It is a common practice to organize a program into three separate files. The class declarations are placed in a header file and the definitions of member functions go into another file. This approach enables the programmer to separate the abstract specification of the interface (class definition) from the implementation details (member functions

Figures

Figures are used exhaustively in the text to illustrate the concepts and methods described.

OBJECTS AS ARGUMENTS

```
#include <iostream>

using namespace std;

class time
{
        int  hours;
        int  minutes;
  public:
        void gettime(int h, int m)
        {  hours = h; minutes = m; }
        void puttime(void)
        {
              cout << hours << " hours and ";
              cout << minutes << " minutes " << "\n";
        }
        void sum(time, time);  // declaration with objects as arguments
};
void time :: sum(time t1, time t2)          // t1, t2 are objects
{
        minutes = t1.minutes + t2.minutes;
        hours = minutes/60;
        minutes = minutes%60;
        hours = hours + t1.hours + t2.hours;
}
int main()
{
        time T1, T2, T3;

        T1.gettime(2,45);    // get T1
        T2.gettime(3,30);    // get T2

        T3.sum(T1,T2);// T3=T1+T2

        cout << "T1 = "; T1.puttime();       // display T1
        cout << "T2 = "; T2.puttime();       // display T2
        cout << "T3 = "; T3.puttime();       // display T3

        return 0;
}
```

PROGRAM 5.7

Program Codes

Codes with comments are given throughout the book to elaborate how the various lines of code work.

The output of Program 2.3 is:

```
Enter Name: Ravinder
Enter Age:  30

Name: Ravinder
Age:  30
```

note

cin can read only one word and therefore we cannot use names with blank spaces.

The program defines **person** as a new data of type class. The class person includes two basic data type items and two functions to operate on that data. These functions are called **member functions**. The main program uses **person** to declare variables of its type. As pointed out earlier, class variables are known as *objects*. Here, p is an object of type **person**. Class objects are used to invoke the functions defined in that class. More about classes and objects is discussed in Chapter 5.

Notes

Language tips and other special considerations are highlighted as notes wherever essential.

SUMMARY

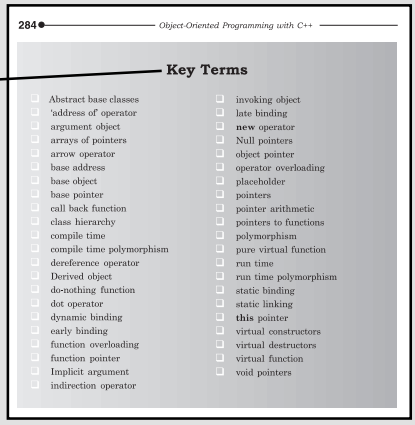
- ⇔ C++ is a superset of C language.
- ⇔ C++ adds a number of object-oriented features such as objects, inheritance, function overloading and operator overloading to C. These features enable building of programs with clarity, extensibility and ease of maintenance.
- ⇔ C++ can be used to build a variety of systems such as editors, compilers, databases, communication systems, and many more complex real-life application systems.
- ⇔ C++ supports interactive input and output features and introduces a new comment symbol // that can be used for single line comments. It also supports C-style comments.
- ⇔ Like C programs, execution of all C++ programs begins at **main( )** function and ends at **return( )** statement. The header file **iostream** should be included at the beginning of all programs that use input/output operations.

Summary

Summary gives the essence of each chapter in the form of bulleted points which will be helpful for a quick review during the examinations.

Key Terms

Key terms listed in each chapter gives the list of important terms discussed in the chapter.



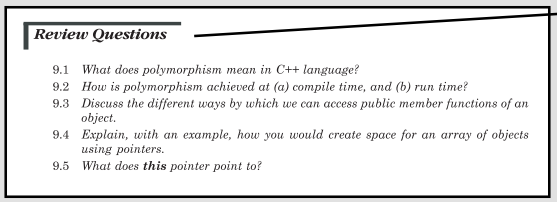
284 Object-Oriented Programming with C++

Key Terms

- Abstract base classes
- 'address of' operator
- argument object
- arrays of pointers
- arrow operator
- base address
- base object
- base pointer
- call back function
- class hierarchy
- compile time
- compile time polymorphism
- dereference operator
- Derived object
- do-nothing function
- dot operator
- dynamic binding
- early binding
- function overloading
- function pointer
- Implicit argument
- indirection operator
- invoking object
- late binding
- **new** operator
- Null pointers
- object pointer
- operator overloading
- placeholder
- pointers
- pointer arithmetic
- pointers to functions
- polymorphism
- pure virtual function
- run time
- run time polymorphism
- static binding
- static linking
- **this** pointer
- virtual constructors
- virtual destructors
- virtual function
- void pointers

Review Questions and Exercises

More than 350 chapter-end exercise problems are given for the students to work out and practice. These include review questions, debugging exercises and programming problems.

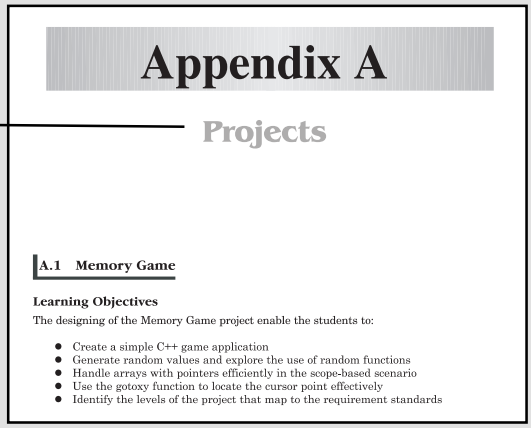


Review Questions

9.1 *What does polymorphism mean in C++ language?*
9.2 *How is polymorphism achieved at (a) compile time, and (b) run time?*
9.3 *Discuss the different ways by which we can access public member functions of an object.*
9.4 *Explain, with an example, how you would create space for an array of objects using pointers.*
9.5 *What does* ***this*** *pointer point to?*

Programming Projects

The two programming projects in Appendix A will give insight on how to integrate the various features of C++ in real-life problems.



Appendix A

Projects

A.1 Memory Game

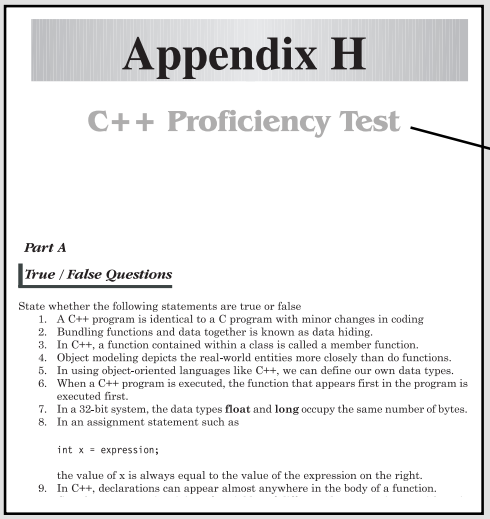
Learning Objectives

The designing of the Memory Game project enable the students to:

- Create a simple C++ game application
- Generate random values and explore the use of random functions
- Handle arrays with pointers efficiently in the scope-based scenario
- Use the gotoxy function to locate the cursor point effectively
- Identify the levels of the project that map to the requirement standards

Proficiency Test

An appendix on Proficiency test will help the readers assess their level of mastery on the language.



Appendix H

C++ Proficiency Test

Part A

True / False Questions

State whether the following statements are true or false

1. A C++ program is identical to a C program with minor changes in coding
2. Bundling functions and data together is known as data hiding.
3. In C++, a function contained within a class is called a member function.
4. Object modeling depicts the real-world entities more closely than do functions.
5. In using object-oriented languages like C++, we can define our own data types.
6. When a C++ program is executed, the function that appears first in the program is executed first.
7. In a 32-bit system, the data types **float** and **long** occupy the same number of bytes.
8. In an assignment statement such as

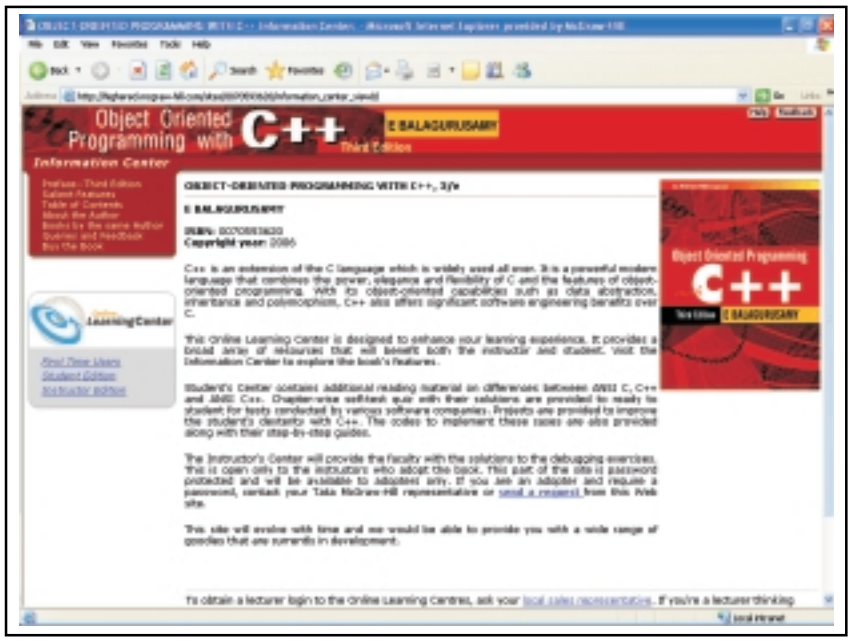
```
int x = expression;
```

   the value of x is always equal to the value of the expression on the right.
9. In C++, declarations can appear almost anywhere in the body of a function.

Web Supplement

The following additional information is available on the web:

- Solution to the *Debugging Exercises*
- Chapter-wise *self-test quiz* with answers
- Complete code with step-by-step description and user manual for the *Employee Attendance Project* discussed in Appendix A
- Differences between ANSI C, C++ and ANSI/ISO C++



Acknowledgements

Since the release of the first edition of this book a decade ago, lakhs of teachers, students and professional programmers have been using the book. Their overwhelming support encouraged me to bring out the Second Edition in 2000 and now the Third Edition. Many of them were kind enough to send their comments and suggestions which have vastly improved the text of the present one. I am greatly indebted to all of them.

My sincere thanks are due to the following reviewers whose suggestions and constructive criticism has helped bring this new edition to its present form.

1. Dr. Vinay Pathak, Department of Computer Science, Hartcourt Butler Technological Institute, Kanpur.

2. Prof. Girish Chandra, Department of Computer Science, Institute of Engineering & Technology, Lucknow.
3. Mr. Shoban Bedagya, Controller, B.Tech, West Bengal University of Technology, Kolkata.
4. Mr. Anindyajyoti Pal, Department of CSE & IT, Heritage Institute of Technology, Kolkata.
5. Prof. Sharat Chandran, Department of CSE, Indian Institute of Technology, Powai, Mumbai.
6. Prof. Rajesh Patwardhan, Computer Science and Information Technology, K.C. College, Mumbai.
7. Prof. R.P. Gohil, Computer Engineering Department, S.V. National Institute of Technology, Surat.
8. Dr. Anil Seth, Department of Computer Engineering, Padre Conceicao College of Engineering, Goa.
9. Dr. T.V. Gopal, Department of Computer Science and Engineering, Anna University, Chennai.
10. Dr. N.V. Subba Reddy, Computer Science Department, Manipal Institute of Technology, Manipal.
11. Mr. Annappa, Department of Computer Engineering, National Institute of Technology, Surathkal.
12. Mr. D. Chouhan, Department of Computer Science and Engineering, Saptagiri College of Engineering, Bangalore.

My sincere thanks are also due to the editorial and publishing professionals of Tata McGraw-Hill for their keen interest and support in bringing out this edition in the present form.

E BALAGURUSAMY