

Preface

In this text we show how object-oriented programming can be used to implement a symbolic algebra system and how the system is applied to different areas in mathematics and physics.

In the most restrictive sense, computer algebra is used for the manipulation of scientific and engineering formulae. Usually, a mathematical formula described in the programming languages such as C, C++ and Java can only be evaluated numerically, by assigning the respective values to each variable. However, the same formula may be treated as a mathematical object in a symbolic algebra system, which allows formal transformation, such as differentiation, integration and series expansion, in addition to the numerical manipulations. This is therefore an indispensable tool for research and scientific computation.

Object-oriented programming has created a new era for programming in computer science as it has been suggested as a possible solution to software development. Basically, object-oriented programming is an important approach to analyzing problems, designing systems and building solutions. By applying this method effectively, the software products become less error prone, easier to maintain, more reusable and extensible.

The purpose of this book is to demonstrate how the features of object-oriented programming may be applied to the development of a computer algebra system. Among the many object-oriented programming languages available nowadays, we have selected C++ as our programming language. It is the most widely used object-oriented programming language, which has been successfully utilized by many programmers in various application areas. The design is based partly on acknowledged principles and partly on solid experience and feedback from actual use. Many experienced individuals and organizations in the industry and academia use C++. In addition to the reasons stated above, we have selected C++ over other object-oriented languages because of its efficiency in execution speed and its utilization of pointers and templates. The Standard Template Library provided by C++ is very helpful for the implementation of a computer algebra system.

Chapter 1 introduces the general notion of Computer Algebra. We discuss the essential properties and requirements of a computer algebra system. Some pitfalls

and limitations are also listed for reference. Finally, we present a computer algebra system — **SymbolicC++**. This new system has many advantages over existing computer algebra systems.

Chapter 2 presents the general mathematics for a computer algebra system. We describe how fundamental mathematical quantities are built up to form more complex mathematical structures.

Chapter 3 gives a brief introduction to some computer algebra systems available in the market place, such as Reduce, Maple, Axiom, Mathematica, MuPAD and Maxima. The basic operations are described for each system. Examples are used to demonstrate the features of these systems.

In Chapter 4, we introduce the language tools in C++ such as the `this` pointer, classes, constructors, and templates. We describe error handling techniques and introduce the concept of exception handling. Examples are also given for demonstration purposes. We also describe recursion. A number of programs illustrate the concepts.

String classes are discussed in Chapter 5. We construct the `String` data type, which serves as a vehicle for introducing the facilities available in C++. The built-in `string` class of C++ is also described in detail. A number of examples show the use of this class. This `string` class of C++ will be used in **SymbolicC++**.

The Standard Template Library (STL) is introduced in Chapter 6 together with a large number of examples. At the core of the Standard Template Library are the three foundational items: containers, algorithm, and iterators. These items work in conjunction with one another. The built-in class in C++ to deal with complex numbers is also introduced. The built-in classes `list`, `vector`, `map`, `complex` will be used in **SymbolicC++**.

Chapter 7 gives a collection of useful classes for computer algebra. We investigate very long integers, rational numbers, quaternions, exact derivatives, vectors, matrices, arrays, bit vectors, finite sets and polynomials. They are the building blocks of mathematics as described in Chapter 2. The internal structures and external interfaces of these classes are described in great detail.

In Chapter 8, we describe how a mathematical expression can be constructed using object-oriented techniques. The computer algebra system **SymbolicC++** is introduced and its internal representations and public interfaces are described. Several examples are also presented to demonstrate the functionalities of the system. A symbolic numeric interface is also described.

In Chapter 9, we apply the classes developed in Chapters 7 and 8 to problems in mathematics and physics. Applications are categorized according to classes. Several classes may be used simultaneously to solve a particular problem. Many interest-

ing problems are presented, such as ghost solutions, Padé approximant, Lie series techniques, Picard's method, Mandelbrot set, etc.

In Chapter 10, we discuss how the programming language Lisp can be used to implement a computer algebra system. We implement an algebraic simplification and differentiation program.

We develop a Lisp system using the object-oriented language C++ in Chapter 11. λ -calculus and its implementation in C++ are also be introduced.

Gene expression programming and its use for numerical and symbolic manipulations is studied in Chapter 12. A number of programs are given to illustrate the technique.

The header files of the classes (abstract data type) introduced in Chapters 8 and 9 are listed in Chapter 13.

The level of presentation is such that one can study the subject early on in one's education in science. There is a balance between practical programming and the underlying language. The book is ideally suited for use in lectures on symbolic computation and object-oriented programming. The beginner will also benefit from the book.

The reference list gives a collection of textbooks useful in the study of the computer language C++ [9], [16], [28], [34], [39], [47], [59]. For data structures we refer to Budd (1994) [11]. For applications in science we refer to Steeb et al. (1993) [49], Steeb (1994) [50], Steeb (2005) [54] and Steeb et al. (2004) [55].

The C++ programs have been tested with all newer C++ compilers which comply with the C++ Standard and include an implementation of the Standard Template Library.

All programs and header files of **SymbolicC++** fall under the GNU General Public License. We omit the following comment (or its equivalent) in all header file and program file listings in the interest of brevity:

```

/*
  SymbolicC++ : An object oriented computer algebra system written in C++

  Copyright (C) 2008 Yorick Hardy and Willi-Hans Steeb

  This program is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation; either version 2 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
  GNU General Public License for more details.

```

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

*/

The license is included with **SymbolicC++** which is available from the web site of the International School for Scientific Computing as described below.

Without doubt, this book can be extended. If you have comments or suggestions, we would be pleased to have them. The email addresses of the authors are:

Yorick Hardy: yhardy@uj.ac.za
 yorickhardy@gmail.com
Willi-Hans Steeb: whsteeb@uj.ac.za
 steebwilli@gmail.com

SymbolicC++ was developed by the International School for Scientific Computing. The web pages of the International School for Scientific Computing are

<http://issc.uj.ac.za/>

The web page also provides the header files for **SymbolicC++**.

Johannesburg, Singapore,
March 2008

Yorick Hardy
Kiat Shi Tan
Willi-Hans Steeb