

Bjarne Stroustrup

BIBLIOTHEQUE DU CERIST



**PROGRAMMING
LANGUAGE**

SECOND EDITION

The C++ Programming Language

Second Edition

Bjarne Stroustrup

AT&T Bell Laboratories
Murray Hill, New Jersey



ADDISON-WESLEY PUBLISHING COMPANY

Reading, Massachusetts • Menlo Park, California • New York
Don Mills, Ontario • Wokingham, England • Amsterdam • Bonn
Sydney • Singapore • Tokyo • Madrid • San Juan • Milan • Paris

Library of Congress Cataloging-in-Publication Data

Stroustrup, Bjarne.

The C++ programming language / Bjarne Stroustrup. -- 2nd ed.

p. cm.

Includes bibliographical references and index.

ISBN 0-201-53992-6

1. C++ (Computer program language) I. Title. II. Title: C plus
plus programming language.

QA76.73.C15S79 1991

005.13'3--dc20

91-27307
CIP

5560



Reprinted with corrections December, 1991

Copyright © 1991 by AT&T Bell Telephone Laboratories, Incorporated.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

This book was typeset in Times and Courier by the author, using a Linotronic 200P phototypesetter and a DEC VAX 8550 running the 10th edition of the UNIX operating system.

DEC, PDP, and VAX are trademarks of Digital Equipment Corporation. UNIX is a registered trademark of AT&T Bell Laboratories.

3 4 5 6 7 8 9 10-HA-9594939291

IST 2 193

Preface

The road goes ever on and on.
— Bilbo Baggins

As promised in the first edition of this book, C++ has been evolving to meet the needs of its users. This evolution has been guided by the experience of users of widely varying backgrounds working in a great range of application areas. The C++ user-community has grown a hundredfold during the six years since the first edition of this book; many lessons have been learned, and many techniques have been discovered and/or validated by experience. Some of these experiences are reflected here.

The primary aim of the language extensions made in the last six years has been to enhance C++ as a language for data abstraction and object-oriented programming in general and to enhance it as a tool for writing high-quality libraries of user-defined types in particular. A “high-quality library,” is a library that provides a concept to a user in the form of one or more classes that are convenient, safe, and efficient to use. In this context, *safe* means that a class provides a specific type-safe interface between the users of the library and its providers; *efficient* means that use of the class does not impose significant overheads in run-time or space on the user compared with hand-written C code.

This book presents the complete C++ language. Chapters 1 through 10 give a tutorial introduction; Chapters 11 through 13 provide a discussion of design and software development issues; and, finally, the complete C++ reference manual is included. Naturally, the features added and resolutions made since the original edition are integral parts of the presentation. They include refined overloading resolution, memory management facilities, and access control mechanisms, type-safe linkage, `const` and `static` member functions, abstract classes, multiple inheritance, templates, and exception handling.

C++ is a general-purpose programming language; its core application domain is

systems programming in the broadest sense. In addition, C++ is successfully used in many application areas that are not covered by this label. Implementations of C++ exist from some of the most modest microcomputers to the largest supercomputers and for almost all operating systems. Consequently, this book describes the C++ language itself without trying to explain a particular implementation, programming environment, or library.

This book presents many examples of classes that, though useful, should be classified as “toys.” This style of exposition allows general principles and useful techniques to stand out more clearly than they would in a fully elaborated program, where they would be buried in details. Most of the useful classes presented here, such as linked lists, arrays, character strings, matrices, graphics classes, associative arrays, etc., are available in “bulletproof” and/or “goldplated” versions from a wide variety of commercial and non-commercial sources. Many of these “industrial strength” classes and libraries are actually direct and indirect descendants of the toy versions found here.

This edition provides a greater emphasis on tutorial aspects than did the first edition of this book. However, the presentation is still aimed squarely at experienced programmers and endeavors not to insult their intelligence or experience. The discussion of design issues has been greatly expanded to reflect the demand for information beyond the description of language features and their immediate use. Technical detail and precision have also been increased. The reference manual, in particular, represents many years of work in this direction. The intent has been to provide a book with a depth sufficient to make more than one reading rewarding to most programmers. In other words, this book presents the C++ language, its fundamental principles, and the key techniques needed to apply it. Enjoy!

Acknowledgments

In addition to the people mentioned in the acknowledgements section in the preface to the first edition, I would like to thank Al Aho, Steve Buroff, Jim Coplien, Ted Goldstein, Tony Hansen, Peter Juhl, Brian Kernighan, Andrew Koenig, Bill Leggett, Lorraine Mingacci, Warren Montgomery, Mike Mowbray, Rob Murray, Jonathan Shapiro, Mike Vilot, and Peter Weinberger for commenting on draft chapters of this second edition. Many people influenced the development of C++ from 1985 to 1991. I can mention only a few: Andrew Koenig, Brian Kernighan, Doug McIlroy, and Jonathan Shapiro. Also thanks to the many participants of the “external reviews” of the reference manual drafts and to the people who suffered through the first year of X3J16.

Murray Hill, New Jersey

Bjarne Stroustrup

Preface to the first Edition

*Language shapes the way we think,
and determines what we can think about.*
— B.L. Whorf

C++ is a general purpose programming language designed to make programming more enjoyable for the serious programmer. Except for minor details, C++ is a superset of the C programming language. In addition to the facilities provided by C, C++ provides flexible and efficient facilities for defining new types. A programmer can partition an application into manageable pieces by defining new types that closely match the concepts of the application. This technique for program construction is often called *data abstraction*. Objects of some user-defined types contain type information. Such objects can be used conveniently and safely in contexts in which their type cannot be determined at compile time. Programs using objects of such types are often called *object based*. When used well, these techniques result in shorter, easier to understand, and easier to maintain programs.

The key concept in C++ is *class*. A class is a user-defined type. Classes provide data hiding, guaranteed initialization of data, implicit type conversion for user-defined types, dynamic typing, user-controlled memory management, and mechanisms for overloading operators. C++ provides much better facilities for type checking and for expressing modularity than C does. It also contains improvements that are not directly related to classes, including symbolic constants, inline substitution of functions, default function arguments, overloaded function names, free store management operators, and a reference type. C++ retains C's ability to deal efficiently with the fundamental objects of the hardware (bits, bytes, words, addresses, etc.). This allows the user-defined types to be implemented with a pleasing degree of efficiency.

C++ and its standard libraries are designed for portability. The current implementation will run on most systems that support C. C libraries can be used from a C++

program, and most tools that support programming in C can be used with C++.

This book is primarily intended to help serious programmers learn the language and use it for nontrivial projects. It provides a complete description of C++, many complete examples, and many more program fragments.

Acknowledgments

C++ could never have matured without the constant use, suggestions, and constructive criticism of many friends and colleagues. In particular, Tom Cargill, Jim Coplien, Stu Feldman, Sandy Fraser, Steve Johnson, Brian Kernighan, Bart Locanthi, Doug McIlroy, Dennis Ritchie, Larry Rosler, Jerry Schwarz, and Jon Shopiro provided important ideas for development of the language. Dave Presotto wrote the current implementation of the stream I/O library.

In addition, hundreds of people contributed to the development of C++ and its compiler by sending me suggestions for improvements, descriptions of problems they had encountered, and compiler errors. I can mention only a few: Gary Bishop, Andrew Hume, Tom Karzes, Victor Milenkovic, Rob Murray, Leonie Rose, Brian Schmult, and Gary Walker.

Many people have also helped with the production of this book, in particular, Jon Bentley, Laura Eaves, Brian Kernighan, Ted Kowalski, Steve Mahaney, Jon Shopiro, and the participants in the C++ course held at Bell Labs, Columbus, Ohio, June 26-27, 1985.

Murray Hill, New Jersey

Bjarne Stroustrup



Contents

Preface	iii
Acknowledgments	iv
Preface to First Edition	v
Acknowledgments	vi
Contents	vii
Notes to the Reader	1
The Structure of This Book	1
Implementation Notes	2
Exercises	3
Design Notes	3
Historical Note	4
C and C++	6
Efficiency and Structure	6
Philosophical Note	8
Thinking about Programming in C++	8
Rules of Thumb	10
Note to C Programmers	10
References	11

A Tour of C++	13
1.1 Introduction	13
1.2 Programming Paradigms	14
1.3 "A Better C"	22
1.4 Support for Data Abstraction	30
1.5 Support for Object-Oriented Programming	36
1.6 Limits to Perfection	41
Declarations and Constants	43
2.1 Declarations	43
2.2 Names	48
2.3 Types	48
2.4 Literals	64
2.5 Named Constants	68
2.6 Saving Space	70
2.7 Exercises	73
Expressions and Statements	75
3.1 A Desk Calculator	75
3.2 Operator Summary	88
3.3 Statement Summary	100
3.4 Comments and Indentation	104
3.5 Exercises	106
Functions and Files	109
4.1 Introduction	109
4.2 Linkage	110
4.3 Header Files	112
4.4 Linkage to Non-C++ Code	119
4.5 How to Make a Library	121
4.6 Functions	123
4.7 Macros	138
4.8 Exercises	140
Classes	143
5.1 Introduction and Overview	143
5.2 Classes and Members	144

5.3 Interfaces and Implementations	153
5.4 Minor Class Features	161
5.5 Construction and Destruction	170
5.6 Exercises	178

Derived Classes **181**

6.1 Introduction and Overview	181
6.2 Derived Classes	182
6.3 Abstract Classes	191
6.4 A Complete Program	193
6.5 Multiple Inheritance	201
6.6 Access Control	211
6.7 Free Store	215
6.8 Exercises	222

Operator Overloading **225**

7.1 Introduction	225
7.2 Operator Functions	226
7.3 User-defined Type Conversion	229
7.4 Literals	236
7.5 Large Objects	236
7.6 Assignment and Initialization	237
7.7 Subscripting	240
7.8 Function Call	242
7.9 Dereferencing	244
7.10 Increment and Decrement	246
7.11 A String Class	248
7.12 Friends and Members	251
7.13 Caveat	252
7.14 Exercises	253

Templates **255**

8.1 Introduction	255
8.2 A Simple Template	256
8.3 List Templates	259
8.4 Function Templates	270
8.5 Template Function Overloading Resolution	277
8.6 Template Arguments	279
8.7 Derivation and Templates	281

Contents

3.8 An Associative Array	284
3.9 Exercises	291

Exception Handling 293

9.1 Error Handling	293
9.2 Discrimination of Exceptions	297
9.3 Naming of Exceptions	300
9.4 Resource Acquisition	308
9.5 Exceptions that are not Errors	315
9.6 Interface Specifications	317
9.7 Uncaught Exceptions	320
9.8 Error-Handling Alternatives	321
9.9 Exercises	324

Streams 325

10.1 Introduction	325
10.2 Output	327
10.3 Input	330
10.4 Formatting	337
10.5 Files and Streams	350
10.6 C Input/Output	356
10.7 Exercises	358

Design and Development 361

11.1 Introduction	361
11.2 Aims and Means	364
11.3 The Development Process	367
11.4 Management	382
11.5 Rules of Thumb	387
11.6 Annotated Bibliography	388

Design and C++ 391

12.1 Design and Programming Language	391
12.2 Classes	402
12.3 Components	422
12.4 Interfaces and Implementations	425
12.5 Rules of Thumb	427

Design of Libraries	429
13.1 Introduction	429
13.2 Concrete Types	431
13.3 Abstract Types	434
13.4 Node Classes	439
13.5 Run-time Type Information	442
13.6 Fat Interfaces	452
13.7 Application Frameworks	455
13.8 Interface Classes	457
13.9 Handle Classes	460
13.10 Memory Management	465
13.11 Exercises	474
 Reference Manual	 477
r.1 Introduction	477
r.2 Lexical Conventions	478
r.3 Basic Concepts	482
r.4 Standard Conversions	488
r.5 Expressions	491
r.6 Statements	508
r.7 Declarations	515
r.8 Declarators	526
r.9 Classes	541
r.10 Derived Classes	554
r.11 Member Access Control	563
r.12 Special Member Functions	570
r.13 Overloading	585
r.14 Templates	595
r.15 Exception Handling	601
r.16 Preprocessing	606
r.17 Appendix A: Grammar Summary	614
r.18 Appendix B: Compatibility	626
 Index	 633