


C++ MATH CLASS LIBRARY

*Permutations, Partitions,
Calculators & Gaming*



MATH
CLASS
LIBRARY
INCLUDED

Scott N. Gerard

C++ Math Class Library

**Permutations, Partitions,
Calculators, and Gaming**



JOHN WILEY & SONS, INC.

New York Chichester Brisbane Toronto Singapore

Associate Publisher: Katherine Schowalter
Editor: Diane Cerra
Managing Editor: Frank Grazioli
Editorial Production & Design: Lachina Publishing Services

This text is printed on acid-free paper.

Copyright© 1994 by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in Initial Capital or ALL CAPITAL LETTERS. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional service. If legal advice or other expert assistance is required, the services of a competent professional person should be sought. FROM A DECLARATION OF PRINCIPLES JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permission Department, John Wiley & Sons, Inc.

Library of Congress Cataloging-in-Publication Data:

Gerard, Scott N., 1956-

C++ math class library : permutations, partitions, calculators &
gaming / by Scott N. Gerard.
p. cm.

Includes bibliographical references and index.

ISBN 0-471-59243-9 (acid-free)

1. C++ (Computer program language) 2. Mathematics—Data
processing. I. Title. II. Title: C plus plus math class library.

QA76.73.C153G47 1994

510 ' .285 ' 5133—dc20

93-29732

CIP

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

P R E F A C E

This book is for programmers who are interested in mathematics and familiar with C++. It is for programmers who are tired of reinventing code that acts “kinda-like” an existing, well-known concept. It is a book for programmers who know that mathematics is full of well-known and useful concepts and who want to tap into these concepts and their power. It is a book for programmers who want to pick up existing code, plug it into their applications, and increase their productivity.

One of the best ways to be more productive is to *stop* writing code and reuse existing programs, particularly for well-defined problems that can be, or have been, solved once and for all. We need libraries of ready-to-run routines.

All of us have written many subroutines in the past. Why not just take all of those routines and tie them up with a pretty bow and call it a library? If only things were that easy. Most subroutines are written with too many assumptions and hooks into their main program. There is a natural tendency to do this. But all those dependencies force us to spend time duplicating their environment or tweaking the code.

Writing general-purpose code is more difficult than writing single-purpose code. Instead of having the exact details of the main program in front of us, we must try to anticipate any and all possible clients. Instead of providing only the function and options required for the specific task at hand, we must provide all relevant operations.

I believe object-oriented (OO) programming makes it easier to write general-purpose code than in non-OO paradigms. Instead of focusing on some hypothetical client and trying to figure out all of its needs, OO focuses on specific objects and asks what operations make sense for that object. OO is not PP (panacea-programming). But although OO has been “overhyped” recently, it does encourage a way of thinking about problems that I believe will be as much a part of future languages as strong data typing has become. Simply put, OO is a good idea and it helps organize programming.

My goals for this book are to

- provide C++ source code for ready-to-run classes,
- leverage the power of mathematical concepts that have stood the test of time,
- give you some ideas where you might apply these concepts in programming applications, and
- teach you some interesting mathematics along the way.

For those of you who like to know the “why” as well as the “what,” some proofs are included. But the proofs are clearly marked and can be skipped.

Mid-level Functions

What kinds of routines should be in a programmer’s library? A library should not contain programs for entire applications. Complete applications are usually too specific to be heavily reused. And the number of distinct applications is far too large for any reasonable library. Complete applications are just too large for inclusion in a library.

Many books have been published with titles like “A Zillion Little Programs for Your PC.” These books are filled with routines to turn on and off the PC speaker, switch video display modes, and so on. These routines fill a certain niche, but are usually not portable to other machines. Primitive functions are needed to write more complex programs, but they do not provide very much function by themselves. They will account for only a small percentage of code in your applications. Therefore, they are too small to greatly increase your productivity because you still have a lot of code to write.

The library routines we’re looking for should be not too small and not too big. A library should be made up of medium-sized routines in terms of both size and complexity. Library routines should be complex enough that they can completely take over all processing in one area of an application. And this suggests writing library routines as objects that know how to maintain themselves.

Unusual Classes

In almost every new programming book—regardless of language—the authors present the ever-popular stacks and queues. Stacks and queues are good examples to illustrate the concepts of data hiding, abstraction, and encapsulation. They are small enough that they can be presented without going through a lot of code, and they are truly useful. However, I will assume you already have a number of these books and therefore do not need yet another version.

Instead, I intend to provide other useful data types that are “above” these basic types, that is, data types that are more complex and provide richer function. These routines are portable (or nearly so) to any machine with a C++ compiler. Some of the classes (in particular, the calculator classes) depend on the lists and sets in Borland’s class library. If you want to compile my classes on a machine without Borland’s classes, you will need to do some reworking.

This book provides a collection of data types that are out of the ordinary, and are intermediate in both size and complexity. There are no stacks or queues, and no sorting routines. All classes are related, in one way or another, to mathematics. This book contains the following classes:

Functions In programming, functions are useful as lookup tables, and for representing finite state machines.

Perm	Permutations have many uses. Besides being a fundamental abstraction of one-to-one and onto functions, they can model card games and other modern puzzles like Rubik's Cube.
Part	Partitions are ideal for representing the concepts of equality and connection.
Polya	This handles unusual types of enumeration problems, like the number of distinct ways to paint the faces of a cube, or the number of distinct bracelets.
Calculator	These classes make it easy to create calculators for your data types.
Region	A region is a mapping between points on a plane, or in a space, and the integers. The region classes support rectangular, triangular, and trapezoidal regions in two dimensions as well as cubical, tetrahedral, pyramidal, and other types of regions in three dimensions.
Xform	These are classes for transforming points including translations, transformations by a group, and general linear transformations.
Hexgrid	These routines manipulate grids of hexagons, which are commonly used in simulation and fantasy games.
Enum IO	These routines read and write enumerations by name.
Hashing	This class combines data into a hash value.
Binomial	In addition to computing binomial coefficients, these routines convert between binary integers and "cogets" (see Chapter 3) in the binomial numbering system.

C++

All the code in this book is in C++ because C++ is the mostly widely used object-oriented language today. This book does *not* cover the basics of the C++ language. There are many good books on C++; for example, see Stroustrup (1991), Lippman (1989), and Meyers (1992).

I will not mount a major defense of the merits of object-oriented programming in C++. If you are reading this book, you probably already agree that C++ is useful, productive, and just plain fun. I will say that I think it is easier to consider all the operations a specific object can reasonably support, than to try and imagine all possible requests from some hypothetical client. This gives me greater confidence that my classes are complete.

Trademarks

All Borland products are trademarks or registered trademarks of Borland International, Inc. Rubik's Cube is a trademark of Ideal Toy Corporation.

Acknowledgments

There are a great many people I want to thank for their help and support of this project. Nancy Hankins, Dave Borrillo, and Craig Orcutt deserve special thanks. They helped

xviii PREFACE

get me started, checked my results, and kept me going. I'd also like to thank Joe and Sue Cahill, Joe Collette, Auther Eberiel, Diane Gerard, Paul Gunsch, Charles and Christian Hankins, Tim Hamel, Jim Herring, Eric Johnson, Steve Knight, Mike Moore, A. Carolyn Neal, Jeff Palm, Curt Rose, Dave and Karen Scudiero, Abolfaz Sirjani, Mike Smith, Francine Stenzel, and Tom Turner.

CONTENTS

PREFACE xv

PART ONE Introduction _____ **1**

1 Getting Started **3**

Minimum Requirements 3

Making a Backup Copy 3

Installation 4

Makefile, 4 • User Assistance and Information, 5

2 Conventions and Notations **6**

OO Diagrams 6

Coding Conventions 8

Naming Conventions, 8 • Consistency Checks, 8

Related Operators 9

Templates, 10 • Ranking, 11 • Virtual Copy Constructors, 12

PART TWO Foundation Modules _____ **15**

3 Binomials **17**

Binomial Coefficients 17

Binomial Iterator 18

Binomial Number System 18

Implementation Comments 19

Possible Improvements 19

SOURCE CODE	20
<i>File BinIter.hpp</i>	20
<i>File Binomial.hpp</i>	22
<i>File Binomial.cpp</i>	24

4 Compare 26

Common Features of the Comparisons	27
Total Compare	29
Partial Compare	29
Set Compare	30
Compare for Numerical Analysis	31

SOURCE CODE	31
<i>File Bool.hpp</i>	31
<i>File TotalCompare.hpp</i>	32
<i>File TotalCompare.cpp</i>	32
<i>File PartCompare.hpp</i>	33
<i>File PartCompare.cpp</i>	34

5 Utility Classes 35

UnInit Class	35
Ident	35
EnumIO	36
IntMath	37
<i>Greatest Common Divisor, 37</i>	• <i>Least Common Multiple, 37</i>
<i>Square Root, 37</i>	
Hashing	37
<i>SimpHasher Class, 38</i>	
Handles and Bodies	39
Implementation Comments	40
<i>Square Root Algorithm, 40</i>	

SOURCE CODE	42
<i>File EnumIO.hpp</i>	42
<i>File EnumIO.cpp</i>	43
<i>File IntMath.hpp</i>	44
<i>File IntMath.cpp</i>	45
<i>File Hasher.hpp</i>	46
<i>File Handle.hpp</i>	49
<i>File Handle.cpp</i>	51

PART THREE Games 53**6 Regions 55**

P2BOARD 55

Point2 Class 57

Region Class 58

Reg2Rect Class 60

Game Boards 61

RegXform Class 62

Xform Class 62

*Xfoldent Class, 63 • XfoXlate Class, 63 • XfoGroup Class, 63
General Linear Transformations, 64*

Three-Dimensional Regions 65

Quad Trees and Oct Trees 65

Usage Notes for Games 68

Implementation Comments 68

Bounds Checking, 68 • Error Handling, 71 • Performance, 71

Possible Extensions 71

SOURCE CODE 73*File Bounds.hpp 73**File Point2.hpp 73**File Point2.cpp 75**File Region.hpp 77**File Region.cpp 80**File Reg2Rect.hpp 81**File Reg2Rect.cpp 82**File RegXSplit.hpp 83**File RegXform.hpp 86**File RegXform.cpp 87**File Xform.hpp 89**File XfoGroup.hpp 90**File XfoGroup.cpp 92**File GroupD4.hpp 93**File GroupD4.cpp 94**File XfoGL2.hpp 96**File XfoGL2.cpp 97**File XfoGL2Imp.hpp 98**File XfoGL2Imp.cpp 100**File P2Board.hpp 102**File P2Board.cpp 102*

7	Trapezoidal Regions	108
	Two-Dimensional Algorithm	108
	<i>Rank, 108 • Number of Cells, 110 • Unrank, 110</i>	
	Reg2Trap Class	112
	Triangle Classes	112
	Row-, Column-, and Diagonal-Orders	112
	Three-Dimensional Algorithm	113
	<i>Number of Cells, 114 • Unrank, 115</i>	
	Implementation Comments	115
	SOURCE CODE	116
	<i>File Reg2Trap.hpp</i>	116
	<i>File Reg2Trap.cpp</i>	117

8	HexGrid	120
	Overview	120
	Hexgrid Coordinates	121
	<i>Natural Coordinates, 122</i>	
	HexPoint Class	122
	H2BOARD	124
	<i>Reg2Hexagon Class, 125 • Reg2Chinese Class, 125</i>	
	<i>Shortest Path, 125</i>	
	A Very Simple Game	127
	Implementation Comments	127
	<i>Shortest Path Proof, 127</i>	
	SOURCE CODE	128
	<i>File HexPoint.hpp</i>	128
	<i>File HexPoint.cpp</i>	132
	<i>File Reg2Hexagon.hpp</i>	135
	<i>File Reg2Hexagon.cpp</i>	138
	<i>File HexGame.cpp</i>	139

PART FOUR Calculation Modules _____ 141

9	Calculator	143
	Starting REALCALC	144
	<i>Literals, 144 • Variables, 144 • Commands, 144</i>	
	<i>Expressions, 145 • Comments, 146 • Line Continuation, 146</i>	
	<i>Interrupting Execution, 146 • Common Commands, 146</i>	
	REALCALC Specific Commands,	148

Implementation Comments	148
<i>Real Class, 148 • Identifier, 149 • Scopes, 151 • SetOfList Class, 152 • Expressions, 153</i>	
Writing Your Own Calculator	157
Possible Extensions	158
SOURCE CODE	159
<i>File CalcIdent.hpp</i>	159
<i>File CalcIdent.cpp</i>	163
<i>File CalcScope.hpp</i>	165
<i>File CalcScope.cpp</i>	169
<i>File RealCalc.cpp</i>	173

10 Functions 177

Background	177
Class Overview	179
<i>DSet Class, 179 • Fun Class, 179 • FunGRng Class, 179</i>	
<i>FunDRng Class, 180 • AutoFun Class, 180 • Perm Class, 180</i>	
<i>Element Class, 181</i>	
Function Calculator	182
<i>C++ Source Code, 183</i>	
DSet Class	186
<i>IdentDSet Class, 186 • ShiftDSet Class, 187 • LinearDSet Class, 187 • ExpDSet Class, 187 • LogDSet Class, 188</i>	
<i>PowerDSet Class, 188</i>	
Component Operations	188
FunDRng Class	189
Function Composition	189
AUTOCALC and AutoFun	190
Element Class	193
Implementation Comments	194
<i>Memory Layout, 194 • Domain and Range Types, 195</i>	
<i>Binary Operators, 195</i>	
Possible Extensions	196
SOURCE CODE	198
<i>File Function.hpp</i>	198
<i>File FunGRng.hpp</i>	204
<i>File Element.hpp</i>	208
<i>File Element.cpp</i>	209

11 Applications of Function 213

- Real Functions 213
- Lookup Tables 214
- Circuit Simulation 214
- Function Composition 215
 - Macro Functions, 215 • Translation Tables, 216 • Color Palettes, 216 • Polynomials, 216*
- Finite State Machines 217
- Semigroups 220
 - Semigroup Representations, 221*

12 Permutations 226

- Background 226
- Input and Output 228
 - Permutation Calculator, 230*
- Interpretations 231
 - Permutations as Data Values, 231 • Permutations as Operators, 232*
 - Other Routines in Class Perm, 233*
- Implementation Comments 234
 - Inversions, 234*
- Possible Improvements 234
- SOURCE CODE 235**
 - File Perm.hpp 235*

13 Permutation Applications 242

- Rearranging 242
 - Randomizing Lists, 242 • Card Games, 242*
 - Change Ringing, 243 • Moving Large Amounts of Data, 244*
 - Round Robin Tournaments, 244 • Group Theory, 246*
 - Symmetries of the Square, 247 • Symmetries of the Cube, 249*
 - Different Representations, 250 • Rubik's Cube, 250 • VLSI Design, 253*
- Possible Improvements 256
- SOURCE CODE 256**
 - File GroupD4P.hpp 256*
 - File GroupD4P.cpp 256*

14 Partitions 260

- Partition Calculator 261
- Equivalence Relations 261
 - Related Relations, 263 • Equivalent Equivalences, 263*
- Matrix Representation 264
- C++ Template Classes 266
 - Constructors, 267 • Assignment, 268 • Partition Literals, 268*
 - Input and Output, 268 • Element Operations, 268 • Partition*
 - Operations, 269 • No Subtraction or Division, 270*
 - Special Partitions, 270*
- Comparing Partitions 271
 - Other Methods, 272*
- Iterators 273
- Partition Laws 273
- Implementation 275
 - General Implementation, 275 • Multiplication Implementation, 276*
 - Proof of Compare, 277 • Ranking, 278 • Random Partitions, 280*
- Possible Extensions 284
 - SOURCE CODE 285**
 - File Part.hpp 285*

15 Partition Applications 294

- Kinds of Examples 294
 - Same Attribute, 294 • Step Functions, 295 • Parallel Equivalence, 295*
- Connection 296
 - Connected Components of a Graph, 296 • Kruskal's Spanning*
 - Tree, 297 • Connected Regions of a Game Board, 297*
 - Connected People over Time, 298 • Connected Continents, 298*
 - FORTTRAN EQUIVALENCE, 299 • Sets as Partitions, 299*
 - Pin Swapping, 300 • Finite State Machines, 301*
 - Polya's Example, 308 • Distinct Circuits, 312*
 - SOURCE CODE 315**
 - File Polya.hpp 315*
 - File Polya.cpp 318*
 - File PolyaT.cpp 321*
- BIBLIOGRAPHY 323
- INDEX 325

FIGURES

2.1.	Modified Booch notation for class and object diagrams	7
2.2.	Example diagram	7
2.3.	Primitive operator $+=$ and derived operator $+$	10
2.4.	Primitive operator $+$ and how operator $+=$ uses it	10
6.1.	Two-dimensional region classes	58
6.2.	Three-dimensional and other region classes	59
6.3.	A quad tree for finding points in the plane	67
6.4.	Two-dimensional boundary checking	69
6.5.	Three-dimensional boundary checking	70
7.1.	Trapezoidal mapping	109
7.2.	General form of the three-dimensional trapezoidal shape	114
8.1.	Hexgrid of hexagons overlaid with a hexgrid of triangles	121
8.2.	Hexgrid with natural coordinates	123
9.1.	REALCALC expression objects	145
9.2.	Identifier classes	149
9.3.	Scope class hierarchy	151
9.4.	SetOfList class hierarchy	153
9.5.	SetOfList objects	153
9.6.	Expression class hierarchy	154
9.7.	Grammar for expression classes	155
10.1.	Domain set class hierarchy	180
10.2.	Function classes	181
11.1.	Finite state machine	218
12.1.	Premultiply versus postmultiply	233
13.1.	Round-robin tournament	245
13.2.	Symmetries of the square	248

13.3.	Symmetries of the cube	249
13.4.	Rubik's Cube	251
14.1.	Partition classes	266
14.2.	Partition implementation	276
14.3.	Partition ranking	279
14.4.	Selecting random partitions	282
15.1.	Finite state machine M1	303
15.2.	Basic partitions for M1	304
15.3.	File m1.fsm	305
15.4.	File m1.peq	305
15.5.	Hasse diagram of closed partitions	306
15.6.	FSM M1min	307
15.7.	Benzene ring	309
15.8.	Bracelet with five beads (slots)	310
15.9.	Distinct bracelets	313
15.10.	Pattern of three input bits	314

PART ONE

Introduction