MICHAEL McMILLAN

Data Structures and Algorithms Using Visual Basic.NET

CAMBRIDGE

DATA STRUCTURES AND ALGORITHMS USING VISUAL BASIC.NET

This is the first Visual Basic.NET (VB.NET) book to provide a comprehensive discussion of the major data structures and algorithms. Here, instead of having to translate material on C++ or Java, the professional or student VB.NET programmer will find a tutorial on how to use data structures and algorithms and a reference for implementation using VB.NET for data structures and algorithms from the .NET Framework Class Library as well as those that must be developed by the programmer.

In an object-oriented fashion, the author presents arrays and ArrayLists, linked lists, hash tables, dictionaries, trees, graphs, and sorting and searching as well as more advanced algorithms, such as probabilistic algorithms and dynamic programming. His approach is very practical, for example using timing tests rather than Big O analysis to compare the performance of data structures and algorithms.

This book can be used in both beginning and advanced computer programming courses that use the VB.NET language and, most importantly, by the professional Visual Basic programmer.

Michael McMillan is Instructor of Computer Information Systems at Pulaski Technical College. With more than twenty years of experience in the computer industry, he has written numerous articles for trade journals such as *Software Development* and *Windows NT Systems*. He is the author of *Perl from the Ground Up* and *Object-Oriented Programming with Visual Basic.Net* and coauthor of several books.

Data Structures and Algorithms Using Visual Basic.NET

Michael McMillan
Pulaski Technical College

Cambridge
University Press

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521547659

First published in print format 2005

ISBN-13 978-0-511-11366-6 eBook (NetLibrary)
ISBN-10 0-511-11366-8 eBook (NetLibrary)

ISBN-13 978-0-521-54765-9 paperback
ISBN-10 0-521-54765-2 paperback

Contents

Preface

The Visual Basic.NET (VB.NET) programming language is not usually associated with the study of data structures and algorithms. The primary reason for this must be because most university and college computer science departments don't consider VB.NET to be a "serious" programming language that can be used to study serious topics. This is primarily a historical bias based on Basic's past as a "nonprogrammer's" language often taught to junior high, senior high, and liberal arts college students, but not to computer science or computer engineering majors.

The present state of the language, however, aligns it with other, more serious programming languages, most specifically Java. VB.NET, in its current form, contains everything expected in a modern programming language, from true object-oriented features to the .NET Framework library, which rivals the Java libraries in both depth and breadth.

Included in the .NET Framework library is a set of collection classes, which range from the Array, ArrayList, and Collection classes, to the Stack and Queue classes, to the Hashtable and the SortedList classes. Students of data structures and algorithms can now see how to use a data structure before learning how to implement it. Previously, an instructor had to discuss the concept of, say, a stack, abstractly until the complete data structure was constructed. Instructors can now show students how to use a stack to perform some computations, such as number base conversions, demonstrating the utility of the data structure immediately. With this background, students can then go back and learn the fundamentals of the data structure (or algorithm) and even build their own implementation.

This book is written primarily as a practical overview of the data structures and algorithms all serious computer programmers need to know and understand. Given this, there is no formal analysis of the data structures and algorithms covered in the book. Hence, there is not a single mathematical formula and not one mention of Big O analysis (for the latter the reader is referred to any of the books listed in the bibliography). Instead, the various data structures and algorithms are presented as problem-solving tools. We use simple timing tests to compare the performance of the data structures and algorithms discussed in the book.

Prerequisites

There are very few prerequisites for this book. The reader should be competent in one or more programming languages, preferably VB.NET, though a course or two in Java will serve as well. C/C++ programmers should not have too much trouble picking up the language. There are no mathematical prerequisites since we don't take a formal approach in the book.

Chapter-by-Chapter Organization

The Introduction provides an overview of object-oriented programming using VB.NET and introduces the benchmark tool used for comparing the performance of the data structures and algorithms studied in the book. This tool is a Timing class developed by the author as a practical means for timing code in the .NET environment.

Chapter 1 introduces the reader to the concept of the data structure as a collection of data. The concepts of linear and nonlinear collections are introduced. The Collection class is demonstrated.

Chapter 2 provides a review of how arrays are constructed in VB.NET, along with demonstrating the features of the Array class. The Array class encapsulates many of the functions associated with arrays (UBound, LBound, and so on) into a single package. Arraylists are special types of arrays that provide dynamic resizing capabilities.

Chapter 3 gives an introduction to the basic sorting algorithms, such as the bubble sort and the insertion sort, and Chapter 4 examines the most fundamental algorithms for searching memory, the sequential and binary searches.

Two classic data structures are examined in Chapter 5—the stack and the queue. This chapter emphasizes the practical use of these data structures in solving everyday problems in data processing. Chapter 6 covers the BitArray class, which can be used to efficiently represent a large number of integer values, such as test scores.

Strings are not usually covered in a data structures book, but Chapter 7 covers strings, the String class, and the StringBuilder class. We feel that because so much data processing in VB.NET is performed on strings, the reader should be exposed to the special techniques found in the two classes. Chapter 8 examines the use of regular expressions for text processing and pattern matching. Regular expressions often provide more power and efficiency than can be had with more traditional string functions and methods.

Chapter 9 introduces the reader to the use of dictionaries as data structures. Dictionaries, and the different data structures based on them, store data as key/value pairs. This chapter shows the reader how to create his or her own classes based on the DictionaryBase class, which is an abstract class. Chapter 10 covers hash tables and the Hashtable class, which is a special type of dictionary that uses a hashing algorithm for storing data internally.

Another classic data structure, the linked list, is covered in Chapter 11. Linked lists are not as important a data structure in VB.NET as they are in a pointer-based language such as C++, but they still play a role in VB.NET programming. Chapter 12 introduces the reader to yet another classic data structure—the binary tree. A specialized type of binary tree, the binary search tree, comprises the primary topic of the chapter. Other types of binary trees are covered in Chapter 15.

Chapter 13 shows the reader how to store data in sets, which can be useful in situations when only unique data values can be stored in the data structure. Chapter 14 covers more advanced sorting algorithms, including the popular and efficient QuickSort, which forms the basis for most of the sorting procedures implemented in the .NET Framework library. Chapter 15 looks at three data structures that prove useful for searching when a binary search tree is not called for: the AVL tree, the red–black tree, and the skip list.

Chapter 16 discusses graphs and graph algorithms. Graphs are useful for representing many different types of data, especially networks. Finally, Chapter 17 introduces the reader to what are really algorithm design techniques—dynamic algorithms and greedy algorithms.

Acknowledgments

There are several different groups of people who must be thanked for helping me finish this book. First, I owe thanks to a certain group of students who

first sat through my lectures on developing data structures and algorithms in VB.NET. These students include (not in any particular order): Matt Hoffman, Ken Chen, Ken Cates, Jeff Richmond, and Gordon Caffey. Also, one of my fellow instructors at Pulaski Technical College, Clayton Ruff, sat through many of the lectures and provided excellent comments and criticism. I also have to thank my department chair, David Durr, for providing me with an excellent environment for researching and writing. I also need to thank my family for putting up with me while I was preoccupied with research and writing. Finally, I offer many thanks to my editor at Cambridge, Lauren Cowles, for putting up with my many questions and topic changes, and her assistant, Katie Hew, who made the publication of this book as smooth a process as possible.

Introduction

In this preliminary chapter, we introduce a couple of topics we'll be using throughout the book. First, we discuss how to use classes and object-oriented programming (OOP) to aid in the development of data structures and algorithms. Using OOP techniques will make our algorithms and data structures more general and easier to modify, not to mention easier to understand.

The second part of this Introduction familiarizes the reader with techniques for performing timing tests on data structures and, most importantly, the different algorithms examined in this book. Running timing tests (also called benchmarking) is notoriously difficult to get exactly right, and in the .NET environment, it is even more complex than in other environments. We develop a Timing class that makes it easy to test the efficiency of an algorithm (or a data structure when appropriate) without obscuring the code for the algorithm or data structures.

Developing Classes

This section provides the reader with a quick overview of developing classes in VB.NET. The rationale for using classes and for OOP in general is not discussed here. For a more thorough discussion of OOP in VB.NET, see McMillan (2004).

One of the primary uses of OOP is to develop user-defined data types. To aid our discussion, and to illustrate some of the fundamental principles of OOP,