



# C++/CLI IN ACTION

Nishant Sivakumar

 MANNING

# contents

---

*preface* xv  
*acknowledgments* xvii  
*about this book* xix

---

## PART 1 THE C++/CLI LANGUAGE ..... 1

---

- 1** ***Introduction to C++/CLI*** 3
- 1.1 The role of C++/CLI 4  
*What C++/CLI can do for you* 6 ▪ *The rationale behind the new syntax* 8
- 1.2 Hello World in C++/CLI 13  
*The /clr compiler option* 15 ▪ *Using VC++ 2005 to create a /clr application* 16
- 1.3 Declaring CLR types 18  
*Class modifiers* 20 ▪ *CLI types and inheritance* 22
- 1.4 Handles: the CLI equivalent to pointers 24  
*Syntax for using handles* 24 ▪ *Tracking references* 26
- 1.5 Instantiating CLI classes 28  
*The gcnew operator* 28 ▪ *Constructors* 31  
*Copy constructors* 33 ▪ *Assignment operators* 36

## 1.6 Boxing and unboxing 38

*Implicit boxing in the new syntax* 38 ▪ *Boxing and type-safety* 40 ▪ *Implementation at the MSIL level* 41  
*Assigning null to a boxed value type* 43

## 1.7 Summary 45

# 2 **Getting into the CLI: properties, delegates and arrays** 46

## 2.1 Properties 47

*Scalar Properties* 48 ▪ *Indexed properties* 55

## 2.2 Delegates and events 58

*Delegates* 59 ▪ *Events* 64

## 2.3 CLI Arrays 68

*Basic CLI array concepts* 69 ▪ *Single-dimensional arrays* 70  
*Multidimensional arrays* 71 ▪ *Jagged arrays* 72  
*Arrays as function arguments* 74 ▪ *Returning arrays from functions* 75 ▪ *Parameter arrays* 76  
*Using System::Array methods* 77 ▪ *Array covariance* 80  
*Arrays of non-CLI objects* 81 ▪ *Directly accessing CLI arrays using native pointers* 83

## 2.4 Summary 84

# 3 **More C++/CLI: stack semantics, function overriding, and generic programming** 86

## 3.1 Stack semantics and deterministic destruction 87

*The new destructor and finalizer syntaxes* 88  
*Stack semantics* 96 ▪ *Guidelines for using destructors and stack semantics* 101

## 3.2 Function overriding 102

*Explicit overriding* 103 ▪ *Renamed overriding* 104  
*Multiple overriding* 105 ▪ *Sealed and abstract functions* 106

## 3.3 Generics and managed templates 108

*Why have parameterized types?* 108 ▪ *Generics syntax for classes and functions* 110 ▪ *Constraint mechanism* 113  
*Issues with the constraint mechanism and simple types* 116  
*Comparison with templates* 120 ▪ *Managed templates* 124

## 3.4 Summary 129

---

**PART 2 MIXING MANAGED AND NATIVE CODE ..... 131**

---

**4 Introduction to mixed-mode programming 133**

- 4.1 Using interior and pinning pointers 135
  - Interior pointers* 136 ■ *Pinning pointers* 141
- 4.2 Working with interop mechanisms 147
  - Accessing a managed library from native code* 148
  - Accessing a native library from managed code* 156
- 4.3 Using mixed types 162
  - Native types with managed members* 162
  - Managed types with native members* 166
- 4.4 Function pointers and delegates: bridging the gap 173
  - Using GetFunctionPointerForDelegate* 173
  - Using GetDelegateForFunctionPointer* 175
- 4.5 Summary 177

**5 Interoperating with native libraries from managed applications 179**

- 5.1 Converting between managed and native types 181
  - Marshalling native strings* 181 ■ *Marshalling arrays* 184
  - Simulating a native static array with managed code* 185
- 5.2 Double thinking in mixed-mode function calls 186
- 5.3 Wrapping a native API and exposing a CLI interface 190
  - Overview of the native API* 191
  - Writing the CLI wrapper* 193
- 5.4 Exposing an MFC extension DLL to .NET 206
  - Overview of the MFC extension DLL* 207
  - Writing the managed regular MFC DLL wrapper* 208
- 5.5 Accessing a COM object via a custom RCW 212
  - The COM object to interop with* 212 ■ *Writing the custom RCW* 215 ■ *Using the custom RCW* 218

- 5.6 Writing a single mixed-mode DLL for both managed and native clients 218
  - Wrapping the System::Object class* 220
  - Writing derived class wrappers* 223
- 5.7 Summary 227

---

## PART 3 USING MANAGED FRAMEWORKS FROM NATIVE APPLICATIONS ..... 229

---

- 6 *Interoping Windows Forms with MFC* 231**
  - 6.1 A simple Windows Forms application 233
  - 6.2 Hosting a Windows Forms control in an MFC dialog 235
  - 6.3 Hosting a Windows Forms control as an MFC view 239
  - 6.4 Giving your MFC apps an Office 2003 style UI 249
  - 6.5 Using a Windows Forms control as an MFC dialog 261
  - 6.6 Using an MFC control in a Windows Forms form 267
    - The custom MFC control* 268 ▪ *Hosting the MFC control from WinForms* 271 ▪ *Using the wrapped control from a WinForms app* 273
  - 6.7 Summary 274
- 7 *Using C++/CLI to target Windows Presentation Foundation applications* 276**
  - 7.1 What is WPF? 278
    - Overview of XAML* 280 ▪ *Anatomy of a simple WPF application* 283
  - 7.2 Using C++/CLI to write a WPF application 288
    - Creating a new C++/CLI Avalon project* 289
    - Using procedural code* 289 ▪ *Dynamically loading XAML* 294 ▪ *Deriving from a class in a C# DLL* 297
  - 7.3 A brief look at some WPF Graphics features 300
    - Using brushes and shapes* 300 ▪ *Transformations* 304

- 7.4 Hosting a WPF control in a native C++ application 310
  - Using a mixed-mode extension DLL* 310
  - Using a mixed-mode application* 319
- 7.5 Hosting a native control in a WPF application 326
- 7.6 Summary 331

## 8 *Accessing the Windows Communication Foundation with C++/CLI* 332

- 8.1 Hello World with the Windows Communication Foundation 334
- 8.2 Duplex communication in WCF 338
  - Creating the service* 340 ■ *Creating the client* 342
- 8.3 Migrating a native DCOM application to WCF 344
  - The example DCOM server* 346 ■ *The native MFC client* 348
  - Writing a WCF proxy service* 351 ■ *Modifying the MFC client to use WCF* 356 ■ *Writing a pure WCF service* 359
  - Comparison of the two migration methods* 362
- 8.4 Hosting a WCF service in an IIS server 362
- 8.5 Summary 366

*Appendix A concise introduction to the .NET Framework* 368  
*index* 385