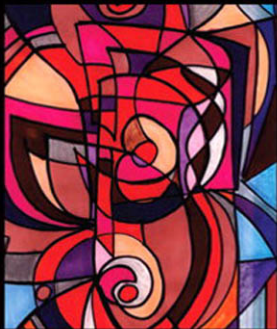


FOREWORDS BY WALTER BRIGHT AND SCOTT MEYERS



The D Programming Language



Andrei Alexandrescu

The D Programming Language

The D Programming Language

Table of Contents

Contents

Foreword

Foreword

Preface

Intended Audience

Organization of the Book

A Brief History

Acknowledgments

1 "D"iving In

1.1 Numbers and Expressions

1.2 Statements

1.3 Function Basics

1.4 Arrays and Associative Arrays

1.4.1 Building a Vocabulary

1.4.2 Array Slicing. Type-Generic Functions. Unit Tests

1.4.3 Counting Frequencies. Lambda Functions

1.5 Basic Data Structures

1.6 Interfaces and Classes

1.6.1 More Statistics. Inheritance

1.7 Values versus References

1.8 Summary

Table of Contents

2 Basic Types. Expressions

2.1 Symbols

2.1.1 Special Symbols

2.2 Literals

2.2.1 Boolean Literals

2.2.2 Integral Literals

2.2.3 Floating-Point Literals

2.2.4 Character Literals

2.2.5 String Literals

2.2.6 Array and Associative Array Literals

2.2.7 Function Literals

2.3 Operators

2.3.1 Lvalues and Rvalues

2.3.2 Implicit Numeric Conversions

2.3.3 Typing of Numeric Operators

2.3.4 Primary Expressions

2.3.5 Postfix Expressions

2.3.6 Unary Expressions

2.3.7 The Power Expression

2.3.8 Multiplicative Expressions

2.3.9 Additive Expressions

2.3.10 Shift Expressions

2.3.11 in Expressions

2.3.12 Comparison Operators

2.3.13 Bitwise OR, XOR, AND

2.3.14 Logical AND

2.3.15 Logical OR

Table of Contents

2.3.16 The Conditional Operator

2.3.17 Assignment Operators

2.3.18 The Comma Operator

2.4 Summary and Quick Reference

3 Statements

3.1 The Expression Statement

3.2 The Compound Statement

3.3 The if Statement

3.4 The static if Statement

3.5 The switch Statement

3.6 The final switch Statement

3.7 Looping Statements

3.7.1 The while Statement

3.7.2 The do-while Statement

3.7.3 The for Statement

3.7.4 The foreach Statement

3.7.5 Foreach on Arrays

3.7.6 The continue and break Statements

3.8 The goto Statement

3.9 The with Statement

3.10 The return Statement

3.11 The throw and try Statements

3.12 The mixin Statement

3.13 The scope Statement

3.14 The synchronized Statement

3.15 The asm Statement

Table of Contents

3.16 Summary and Quick Reference

4 Arrays, Associative Arrays, and Strings

4.1 Dynamic Arrays

- 4.1.1 Length
- 4.1.2 Bounds Checking
- 4.1.3 Slicing
- 4.1.4 Copying
- 4.1.5 Comparing for Equality
- 4.1.6 Concatenating
- 4.1.7 Array-wise Expressions
- 4.1.8 Shrinking
- 4.1.9 Expanding
- 4.1.10 Assigning to .length

4.2 Fixed-Size Arrays

- 4.2.1 Length
- 4.2.2 Bounds Checking
- 4.2.3 Slicing
- 4.2.4 Copying and Implicit Conversion
- 4.2.5 Comparing for Equality
- 4.2.6 Concatenating
- 4.2.7 Array-wise Operations

4.3 Multidimensional Arrays

4.4 Associative Arrays

- 4.4.1 Length
- 4.4.2 Reading and Writing Slots
- 4.4.3 Copying
- 4.4.4 Comparing for Equality

Table of Contents

4.4.5 Removing Elements

4.4.6 Iterating

4.4.7 User-Defined Types as Keys

4.5 Strings

4.5.1 Code Points

4.5.2 Encodings

4.5.3 Character Types

4.5.4 Arrays of Characters + Benefits = Strings

4.6 Arrays' Maverick Cousin: The Pointer

4.7 Summary and Quick Reference

5 Data and Functions. Functional Style

5.1 Writing and unittesting a Simple Function

5.2 Passing Conventions and Storage Classes

5.2.1 Ref Parameters and Returns

5.2.2 In Parameters

5.2.3 Out Parameters

5.2.4 Static Data

5.3 Type Parameters

5.4 Signature Constraints

5.5 Overloading

5.5.1 Partial Ordering of Functions

5.5.2 Cross-Module Overloading

5.6 Higher-Order Functions. Function Literals

5.6.1 Function Literals versus Delegate Literals

5.7 Nested Functions

5.8 Closures

Table of Contents

5.8.1 OK, This Works. Wait, It Shouldn't. Oh, It Does!

5.9 Beyond Arrays. Ranges. Pseudo Members

5.9.1 Pseudo Members and the @property Attribute

5.9.2 Reduce-Just Not ad Absurdum

5.10 Variadic Functions

5.10.1 Homogeneous Variadic Functions

5.10.2 Heterogeneous Variadic Functions

5.11 Function Attributes

5.11.1 Pure Functions

5.11.2 The nothrow Function Attribute

5.12 Compile-Time Evaluation

6 Classes. Object-Oriented Style

6.1 Classes

6.2 Object Names Are References

6.3 It's an Object's Life

6.3.1 Constructors

6.3.2 Forwarding Constructors

6.3.3 Construction Sequence

6.3.4 Destruction and Deallocation

6.3.5 Tear-Down Sequence

6.3.6 Static Constructors and Destructors

6.4 Methods and Inheritance

6.4.1 A Terminological Smörgåsbord

6.4.2 Inheritance Is Subtyping. Static and Dynamic Type

6.4.3 Overriding Is Only Voluntary

6.4.4 Calling Overridden Methods

6.4.5 Covariant Return Types

Table of Contents

6.5 Class-Level Encapsulation with static Members

6.6 Curbing Extensibility with final Methods

6.6.1 Final Classes

6.7 Encapsulation

6.7.1 Private

6.7.2 Package

6.7.3 Protected

6.7.4 Public

6.7.5 Export

6.7.6 How Much Encapsulation?

6.8 One Root to Rule Them All

6.8.1 String toString()

6.8.2 Size_t toHash()

6.8.3 Bool opEquals(Object rhs)

6.8.4 Int opCmp(Object rhs)

6.8.5 Static Object factory(string className)

6.9 Interfaces

6.9.1 The Non-Virtual Interface (NVI) Idiom

6.9.2 Protected Primitives

6.9.3 Selective Implementation

6.10 Abstract Classes

6.11 Nested Classes

6.11.1 Classes Nested in Functions

6.11.2 Static Nested Classes

6.11.3 Anonymous Classes

6.12 Multiple Inheritance

6.13 Multiple Subtyping

Table of Contents

6.13.1 Overriding Methods in Multiple Subtyping Scenarios

6.14 Parameterized Classes and Interfaces

6.14.1 Heterogeneous Translation, Again

6.15 Summary

7 Other User-Defined Types

7.1 Structs

7.1.1 Copy Semantics

7.1.2 Passing struct Objects to Functions

7.1.3 Life Cycle of a struct Object

7.1.4 Static Constructors and Destructors

7.1.5 Methods

7.1.6 Static Members

7.1.7 Access Specifiers

7.1.8 Nesting structs and classes

7.1.9 Nesting structs inside Functions

7.1.10 Subtyping with structs. The @disable Attribute

7.1.11 Field Layout. Alignment

7.2 Unions

7.3 Enumerated Values

7.3.1 Enumerated Types

7.3.2 Enum Properties

7.4 Alias

7.5 Parameterized Scopes with template

7.5.1 Eponymous templates

7.6 Injecting Code with mixin templates

7.6.1 Symbol Lookup inside a mixin

7.7 Summary and Reference

Table of Contents

8 Type Qualifiers

8.1 The immutable Qualifier

8.1.1 Transitivity

8.2 Composing with immutable

8.3 Immutable Parameters and Methods

8.4 Immutable Constructors

8.5 Conversions involving immutable

8.6 The const Qualifier

8.7 Interaction between const and immutable

8.8 Propagating a Qualifier from Parameter to Result

8.9 Summary

9 Error Handling

9.1 Throwing and catching

9.2 Types

9.3 Finally clauses

9.4 Nothrow Functions and the Special Nature of Throwable

9.5 Collateral Exceptions

9.6 Stack Unwinding and Exception-Safe Code

9.7 Uncaught Exceptions

10 Contract Programming

10.1 Contracts

10.2 Assertions

10.3 Preconditions

10.4 Postconditions

Table of Contents

10.5 Invariants

10.6 Skipping Contract Checks. Release Builds

10.6.1 Enforce Is Not (Quite) assert

10.6.2 Assert(false)

10.7 Contracts: Not for Scrubbing Input

10.8 Contracts and Inheritance

10.8.1 Inheritance and in Contracts

10.8.2 Inheritance and out Contracts

10.8.3 Inheritance and invariant Contracts

10.9 Contracts in Interfaces

11 Scaling Up

11.1 Packages and Modules

11.1.1 Import Declarations

11.1.2 Module Searching Roots

11.1.3 Name Lookup

11.1.4 Public import Declarations

11.1.5 Static import Declarations

11.1.6 Selective imports

11.1.7 Renaming in imports

11.1.8 The module Declaration

11.1.9 Module Summaries

11.2 Safety

11.2.1 Defined and Undefined Behavior

11.2.2 The @safe, @trusted, and @system Attributes

11.3 Module Constructors and Destructors

11.3.1 Execution Order within a Module

11.3.2 Execution Order across Modules

Table of Contents

11.4 Documentation Comments

11.5 Interfacing with C and C++

11.6 Deprecated

11.7 Version Declarations

11.8 Debug Declarations

11.9 D's Standard Library

12 Operator Overloading

12.1 Overloading Operators

12.2 Overloading Unary Operators

12.2.1 Using mixin to Consolidate Operator Definitions

12.2.2 Postincrement and Postdecrement

12.2.3 Overloading the cast Operator

12.2.4 Overloading Ternary Operator Tests and if Tests

12.3 Overloading Binary Operators

12.3.1 Operator Overloading [sup(2)]

12.3.2 Commutativity

12.4 Overloading Comparison Operators

12.5 Overloading Assignment Operators

12.6 Overloading Indexing Operators

12.7 Overloading Slicing Operators

12.8 The \$ Operator

12.9 Overloading foreach

12.9.1 Foreach with Iteration Primitives

12.9.2 Foreach with Internal Iteration

12.10 Defining Overloaded Operators in Classes

12.11 And Now for Something Completely Different:

Table of Contents

opDispatch

12.11.1 Dynamic Dispatch with opDispatch

12.12 Summary and Quick Reference

13 Concurrency

13.1 Concurrentgate

13.2 A Brief History of Data Sharing

13.3 Look, Ma, No (Default) Sharing

13.4 Starting a Thread

13.4.1 Immutable Sharing

13.5 Exchanging Messages between Threads

13.6 Pattern Matching with receive

13.6.1 First Match

13.6.2 Matching Any Message

13.7 File Copying-with a Twist

13.8 Thread Termination

13.9 Out-of-Band Communication

13.10 Mailbox Crowding

13.11 The shared Type Qualifier

13.11.1 The Plot Thickens: shared Is Transitive

13.12 Operations with shared Data and Their Effects

13.12.1 Sequential Consistency of shared Data

13.13 Lock-Based Synchronization with synchronized classes

13.14 Field Typing in synchronized classes

13.14.1 Temporary Protection == No Escape

13.14.2 Local Protection == Tail Sharing

Table of Contents

13.14.3 Forcing Identical Mutexes

13.14.4 The Unthinkable: casting Away shared

13.15 Deadlocks and the synchronized Statement

13.16 Lock-Free Coding with shared classes

13.16.1 Shared classes

13.16.2 A Couple of Lock-Free Structures

13.17 Summary

Bibliography

Index