
Table of Contents

| | |
|---|-----------|
| Conventions Used in this Book | xv |
| 1. Prototype Everything | 1 |
| Simple Prototype for the QCMDEXC API | 2 |
| 2. Prototyping a Call to a Program | 3 |
| Use EXTPGM to Prototype Program Calls | 3 |
| Using EXTPGM with an Alternate Prototype Name | 4 |
| 3. An Alternative to QCMDEXC | 5 |
| Prototype for the system() function | 5 |
| 4. Subprocedure-style Entry Parameter List for Programs | 7 |
| Use a Prototype to Replace *ENTRY/PLIST for Programs | 7 |
| 5. Add NOMAIN to the Header Specification of Secondary Modules | 9 |
| Use NOMAIN in all Secondary Modules' Source Members | 9 |
| 6. Monitoring C Function Runtime Errors | 10 |
| Write C Runtime Errors to the Joblog | 11 |
| What about CPF Messages? | 13 |
| RPG IV Declaration for C Runtime CPF Messages | 13 |
| Retrieve CPF Message for C Runtime Functions | 13 |
| 7. Use PSDS to Retrieve Job Information | 15 |
| Job Information from the PSDS — Helper Subprocedures | 16 |
| 8. Use QUALIFIED Data Structures | 17 |
| Traditional Data Structures | 17 |
| Qualified Data Structures | 17 |
| Qualified Syntax Use | 18 |
| 9. Copy Subfields Between Qualified Data Structures | 19 |
| Using EVAL-CORR to Copy a Data Structure | 20 |
| 10. Nested Data Structures | 21 |
| Nesting Data Structures within One Another | 21 |
| 11. *ALL and *ALLX 'xx' — The Repeating Constants | 24 |
| Initialize a Field to "Hex Zeros" with *ALLX '00' | 25 |
| Test a Field for a Repeating Pattern | 25 |

| | |
|---|-----------|
| 12. Embed Compiler Parameters into Source Members | 26 |
| Synchronize Source Statement Numbers with SEU and Debug | 26 |
| Use OPTION(*SRCSTMT) to Sequence Compiler Listings | 27 |
| Reduce Debug Fatigue with *NODEBUGIO | 27 |
| Use OPTION(*NODEBUG) to Avoid F10 Fatigue in Debug | 27 |
| Combine Header Specification Options | 27 |
| Specify Multiple OPTIONS | 27 |
| 13. Avoid “Surprise Initialize” | 29 |
| Use INZ to Fix Decimal Data Errors | 30 |
| 14. Qualified Externally Described Files (1) | 31 |
| Qualified, Externally Described Files | 32 |
| 15. Qualified Externally Described Files (2) | 33 |
| 16. Calculate the End-of-Month Date | 33 |
| Calculate End-of-Month in Fixed Format | 33 |
| Calculate End-of-Month in Free Format | 34 |
| Get End-Of-Month Subprocedure | 34 |
| 17. Using Free-Format Comments in Fixed-Format Code | 35 |
| Free-Format Comments in Fixed Format | 35 |
| 18. Get Day-of-Week Name | 36 |
| GetDayName Subprocedure Source Code | 36 |
| 19. Run CL Commands from an FTP Client | 38 |
| 20. Put Your Program to Sleep | 39 |
| RPG IV Prototype for the sleep() C Runtime Function | 39 |
| 21. Use VARYING to Improve Performance | 40 |
| VARYING Keyword to Help Improve %SCAN Performance | 41 |
| 22. Converting Numeric to Character with %CHAR | 42 |
| Convert Numeric to Character | 42 |
| 23. Converting Character to Numeric | 43 |
| %DEC to Convert Text to Packed Decimal | 43 |
| %INT to Convert Text to Integer | 43 |
| Convert Character to Number Using RPG xTools’s CharToNum | 44 |
| 24. Easier Text Concatenation | 45 |
| Concatenating Character and Numeric Data | 47 |
| Generic Prototype for sprintf() Function | 47 |
| Use sprintf() for Concatenation with Substitution Variables | 48 |

| | |
|---|-----------|
| 25. Create an Auto-Extend User Space | 50 |
| Prototype for QUSCRTUS (Create User Space) API | 50 |
| Change User Space Attributes (QUSCUSAT) API Prototype | 51 |
| Create User Space and Change Attribute to Auto-Extend | 52 |
| Changing Multiple User Space Attributes | 53 |
| User Space Attribute Data Structure Templates | 54 |
| 26. Declare Data Structures as Arrays | 55 |
| Data Structure Arrays — Accessing the Elements | 56 |
| 27. Initialize Fields to Job Date, System Date, or User Profile | 58 |
| *JOB, *SYS, and *USER Initial Values | 58 |
| What about Job Date at Runtime? | 59 |
| 28. Use C Runtime Functions in RPG | 61 |
| Include the QC2LE Binding Directory to Use C Runtime Functions | 61 |
| 29. Compare and Ignore Case | 61 |
| Prototype for memicmp() — Compare and Ignore Case | 61 |
| Compare Two Fields and Ignore Case | 62 |
| 30. Free Online Prototypes for APIs, C Functions, MI Instructions | 63 |
| RPG IV Prototype Source Code Web Address | 63 |
| 31. Checking for Valid Dates with the TEST OpCode | 64 |
| Check a Numeric Field for Valid Date Value | 64 |
| Check a Date Variable for a Valid Date Value | 65 |
| 32. Using the Secret 'X' Edit Code to Convert Numeric to Character | 66 |
| Equivalent MOVEL and EVAL Opcodes Using 'X' Edit Code | 68 |
| 33. %ADDR — Address of a Variable | 69 |
| 34. Understanding API Documentation — Bin(4) Parameters | 71 |
| QUSRTVUS (Retrieve User Space Data) API Documentation | 72 |
| QUSRTVUS (Retrieve User Space Data) Prototype | 72 |
| 35. Understanding API Documentation — Pointer Parameters | 73 |
| QUSPTRUS (Retrieve Pointer to User Space Data) API Documentation | 73 |
| QUSPTRUS (Get Pointer to User Space) Prototype | 73 |
| 36. Better Performance when Accessing User Space Data | 75 |
| QUSPTRUS (Get Pointer to User Space) Prototype | 75 |
| 37. Integer Data-types — More Efficient than Packed Decimal | 77 |
| Signed Integer Sizes and Ranges — RPG Data-type I | 78 |
| Unsigned Integer Sizes and Ranges — RPG Data-type U | 78 |

| | |
|---|------------|
| 38. Sending a Program Message in RPG | 79 |
| Send Program Message (QMHSNDPDM) API Prototype (Subset) | 80 |
| SNDMSG Subprocedure — QMHSNDPDM API Wrapper | 81 |
| 39. Retrieve the Function Key Used on a Display File | 85 |
| Use the INFDS Position 369 to Retrieve the Function Key ID Code | 86 |
| Function Key Scan Codes | 86 |
| 40. Copying More than 64k of Data | 88 |
| Prototype for the C Runtime memcpy() Function | 88 |
| Prototype for the MI CPYBYTES Function | 88 |
| Copy Big Memory Using memcpy() | 89 |
| 41. Use %XFOOT with %LEN | 90 |
| Length vs. Size | 90 |
| Use %XFOOT to Calculate Total Length of All Array Elements | 91 |
| Use %XFOOT with VARYING Array Elements | 91 |
| Use %XFOOT with %SUBARR to Sum Up Lengths of Elements 1 to 2 | 92 |
| 42. Use %SUBARR to Subscript Arrays | 93 |
| Sort 50 Elements of a Large Array | 93 |
| 43. Use EXTFILE to Avoid Needless Overrides | 94 |
| EXTFILE and EXTMBR to Override Runtime File, Library Member | 94 |
| What about Overrides? | 94 |
| EXTFILE Syntax | 94 |
| Using an Initial Value via *INZSR with EXTFILE | 96 |
| Using a Parameter Variable with EXTFILE | 96 |
| Module 1: Set up a Variable for EXPORT | 97 |
| Module 2: Import the File Name Variable | 97 |
| 44. Subprocedure Parameters Rule 1 — Default Behavior | 99 |
| 45. Subprocedure Parameters Rule 2 — Const Parameters | 100 |
| Using CONST on Parameter Definitions | 100 |
| 46. Subprocedure Parameters Rule 3 — VARYING | 102 |
| Using VARYING on Parameter Definitions | 102 |
| Using VARYING and CONST on Parameter Definitions | 102 |
| 47. Subprocedure Parameters Rule 4 — Optional Parameters | 104 |
| Using OPTIONS(*NOPASS) to Declare Optional Parameters | 105 |
| 48. Subprocedure Parameters Rule 5 — Skipping Parameters | 106 |
| Use OPTIONS(*OMIT) to Allow Parameter Skipping | 106 |

| | |
|--|------------|
| Test for an Omitted Parameter | 107 |
| Testing for Omitted Parameters before V5R1 | 108 |
| 49. Data Structure Templates | 109 |
| Use LIKEDS to Use a Data Structure Template | 109 |
| 50. Boolean Assignment | 112 |
| Boolean Assignment | 113 |
| 51. Creating Even-Length Packed Fields | 114 |
| Use PACKEVEN with From/To Column Notation | 115 |
| 52. Sorting Arrays with SubArrays | 116 |
| Sorting an Array by SubArray Name | 116 |
| Cross-footing a SubArray | 117 |
| 53. Convert between Lower- and Uppercase Letters | 118 |
| Prototype for QlgConvertCase API | 119 |
| FRCB_T Data Structure Template for QlgConvertCase API | 119 |
| toLower() and toUpper() Subprocedure Wrappers for QlgConvertCase | 121 |
| 54. Overlapping Data Structures | 124 |
| Overlapping Data Structures | 125 |
| Multi-format Files Implemented with Overlapping Data Structures | 126 |
| Overlapping Data Structures Using BASED Keyword | 127 |
| 55. Dynamic Arrays — Dynamically Allocated Array Elements | 128 |
| Dynamically Allocated Array Elements | 129 |
| Declare the Array with the BASED Keyword | 129 |
| Retrieve a Pointer to a User Space | 129 |
| 56. Converting Date Formats with the QWCCVDT API | 131 |
| Prototype for QWCCVTDT API | 131 |
| Using QWCCVTDT to Convert Date Format | 132 |
| Use QWCCVTDT to Convert Julian Date to Job Date Format | 132 |
| Use QWCCVTDT to YYYYMMDD to MMDDYYYY | 133 |
| QWCCVTDT Data Structures | 133 |
| Data Structures for QWCCVTDT | 133 |
| Data Structure for the Time Zone Info Parameter of QWCCVTDT | 135 |
| 57. Converting Date Formats with the CEExxxx APIs | 136 |
| Converting Non-Standard Date Formats to Real Date Variables | 138 |
| Formatting a Date as Words | 139 |
| 58. Calculated Day of Week — Zeller's Congruence | 140 |
| Zeller's Congruence in RPG IV | 140 |

| | |
|---|------------|
| 59. Calculated Day of Week — API Method | 141 |
| CEEDYWK Prototype | 141 |
| Get Day of Week Prototype | 142 |
| Get Day of Week Procedure Implementation | 142 |
| 60. LIKE Keyword Misbehavior — Zoned to Packed | 144 |
| Demonstrate LIKE Reverting to Packed Data-type | 146 |
| 61. Default Data-type: Not So Consistent | 147 |
| 62. Debugging Variables that Have Debugger Command Names | 149 |
| Display the Contents of a Field Named EVAL in the Debugger | 150 |
| 63. Viewing Field Contents in Hex in Debug | 151 |
| 64. Display the First Few Bytes During Debug | 152 |
| 65. Display Contents of Local Variables with %LOCALVAR | 154 |
| 66. Convert Character to Numeric — Using MI | 155 |
| Prototype for _CVTEFN MI Instruction in RPG IV | 156 |
| Data Structure for _CVTEFN | 157 |
| Named Constants for Data-types Used by _CVTEFN | 157 |
| Using _CVTEFN to Convert Varying Character to Numeric | 158 |
| Using _CVTEFN to Convert Fixed-length Character to Numeric | 159 |
| 67. Converting To and From Hexadecimal | 160 |
| Prototypes for Converting To and From Hexadecimal | 160 |
| 68. Using Decimal Fields as Real-Date Values | 162 |
| Convert an 8-digit Numeric Value to a Real-Date Variable | 163 |
| Convert a Real-Date Value to an 8-digit Numeric Value | 163 |
| Convert a Real-Date Value to a 10-position Character Value | 164 |
| 69. Check Object Existence | 165 |
| QUSEC Data Structure Template | 165 |
| Check if Object Exists Subprocedure | 166 |
| 70. Supporting Qualified Object Syntax | 168 |
| ParseObject() Subprocedure Source | 169 |
| 71. Explained: Bytes Provided, Bytes Available, and Bytes Returned | 171 |
| Bytes Provided — Length of the Data Structure | 171 |
| Bytes Returned | 172 |
| Bytes Available | 173 |
| 72. Converting to/from ASCII and Other Character Sets | 175 |
| The iconv() Conversion Environment Handle | 176 |

| | |
|--|------------|
| The iconv() Conversion Structure | 175 |
| Prototype for QtqIconvOpen — Open iconv() Environment | 177 |
| Prototype for iconv() API | 178 |
| Prototype for the iconv_close() API | 179 |
| <i>Flowchart of the iconv() Conversion Process</i> | 180 |
| Using iconv() for CCSID/Character Conversion — Complete Example | 181 |
| 73. Register an Exit Routine for a Program or Service Program | 182 |
| CEE4RAGE Prototype — Register Activation Group Exit Procedure | 183 |
| CEE4RAGE2 Prototypes with Conditional Directives | 183 |
| Exit Subprocedure Prototype for CEE4RAGE | 183 |
| CEE4RAGE2 — Exit Subprocedure Prototype | 184 |
| User-Written Exit SubProcedure Skeleton | 184 |
| 74. Specifying IFS File Names Correctly | 187 |
| Open an IFS File Name trimmed with %TRIMR | 188 |
| Properly Specifying IFS File Names and Paths | 188 |
| IFS open(), access(), and stat() prototypes with *TRIM | 189 |
| 75. Checking if IFS Files Exist | 190 |
| Prototype for the C Runtime access() Function | 190 |
| Check for IFS File Existence | 191 |
| 76. RC4 Encryption Using Encryption APIs | 192 |
| Qc3EncryptData API Prototype | 192 |
| rc4Encrypt() — Subprocedure to Encrypt Data Using RC4 | 195 |
| 77. Writing Text to the Joblog | 197 |
| Qp0zLprintf API Prototype | 198 |
| Joblog Subprocedure | 199 |
| Writing to the Joblog with the Joblog Subprocedure | 200 |
| 78. Reading Save Files with RPG IV | 201 |
| Copy Save File — Example Program | 202 |
| 79. Encrypting Save Files in RPG IV | 204 |
| Copy Save File with Encryption | 205 |
| Copy and Encrypt Save File Command Definition | 205 |
| Encrypt a Save File | 206 |
| 80. Global and Local Variables | 209 |
| Global and Local Variable Name Conflict | 210 |

| | |
|--|------------|
| 81. Create Source Members Used to Create Service Programs | 211 |
| Source Code for the DATERTN Module | 212 |
| 82. Binder Source for a Service Program | 214 |
| Binder Source with Hard-Coded Signature | 215 |
| Binder Source with New Export | 216 |
| 83. Create Binder Language the Easy Way | 217 |
| Results of a RTVBNSRC Command | 217 |
| Binder Source — A Closer Look | 218 |
| 84. Linking to a Service Program from an RPG IV Program | 219 |
| What Is a Binding Directory? | 220 |
| Specifying a Binding Directory | 220 |
| 85. Swap Bytes — Big Endian to Little Endian in RPG IV | 222 |
| Original Big Endian 32-bit Data | 223 |
| First Step — Swap Byte 1 with 2, and 3 with 4 | 223 |
| Second Step — Swap 2-Byte Words | 223 |
| SwapByte Subprocedure — Convert Big Endian to Little Endian | 224 |
| 86. Dumping the Call Stack with Qp0zDumpStack | 225 |
| Prototype for the Qp0zDumpStack API | 225 |
| Dumping the Call Stack from within a Program | 225 |
| Formatted Call Stack Dump Output | 226 |
| 87. Using Subprocedure Return Values | 227 |
| Define a Subprocedure Return Value | 227 |
| Using a Returned Value | 228 |
| 88. How Does the %EDITC (Edit Code) Built-in Function Work? | 229 |
| 89. Solid Parameter Testing | 231 |
| Testing Parameter Count with %PARMSO | 231 |
| Testing for Skipped Parameters | 233 |
| 90. Create ASCII Text Files on the IFS | 234 |
| Creating an ASCII Text IFS File | 234 |
| Reopening an IFS File as Text | 235 |
| IFS API Prototypes | 235 |
| Figurative Constants Used by IFS APIs | 236 |
| IFS API Prototypes for RPG IV | 236 |
| CrtASCIIFile (Create ASCII File) Subprocedure | 237 |

| | |
|---|------------|
| 91. High-level Math in RPG IV | 239 |
| CEE Math API Prototypes | 240 |
| C Math Library — An Alternative to CEE Math APIs | 241 |
| C Runtime Math Function Prototypes in RPG IV | 241 |
| 92. Program Described Print File with Dynamic Spacing | 244 |
| PRTCTL Printer File Keyword and Data Structure | 244 |
| Using PRTCTL to Control Printed Output | 245 |
| Enhanced PRTCTL Data Structure | 246 |
| 93. Aligning or Centering Text in a Character Field | 248 |
| Left- or Right-Justifying Text | 248 |
| Centering Text | 249 |
| Using the CENTER Text Subprocedure | 250 |
| 94. Debug a Batch Job | 251 |
| A Faster Way to Enter the Job Number | 252 |
| 95. Find and Replace with Regular Expressions | 254 |
| REGCOMP — Compile a Regular Expression String | 255 |
| REGCOMP Control Flags | 255 |
| REGCOMP Return Values | 256 |
| REGEXEC — Search Using Regular Expression | 256 |
| REGFREE — Release the Regular Expression Work Buffer | 256 |
| REGERROR — Retrieve Regular Expression Errors | 256 |
| RegComp() Parameters | 258 |
| RegExec() Parameters | 259 |
| 96. Use DLTOVR when Using OVRDBF OVRSCOPE(*JOB) | 261 |
| OVRDBF OVRSCOPE(*JOB) Corresponds to DLTOVR LVL(*JOB) | 261 |
| 97. Use a FOR Loop to Allow Multiple Exit Points | 262 |
| Traditional Nested Logic | 262 |
| Using a FOR Loop with LEAVE | 263 |
| About the FOR OpCode | 263 |
| 98. Source-level Debugger for Legacy RPG III | 265 |
| 99. Set and Get Environment Variables from within RPG IV | 266 |
| Prototypes for PUTENV and GETENV APIs | 266 |
| Two Environments | 268 |
| CL Access to the Environment | 269 |
| Additional Environment API Prototypes | 270 |
| Environment APIs | 270 |

| | |
|---|------------|
| 100. Simple Scan with Ignore Upper/Lowcase | 272 |
| Use Nested %XLATE Inside %SCAN | 272 |
| 101. Set the CLASSPATH for Java within RPG IV | 274 |
| Retrieve the Existing CLASSPATH | 275 |
| Scan the New CLASSPATH for %CLASSPATH% | 275 |
| Replace %CLASSPATH% with the Old CLASSPATH | 276 |
| Set the CLASSPATH Environment Variable | 277 |
| SetClasspath() Subprocedure Implementation | 278 |
| Epilogue | 279 |
| | |
| <i>Appendix: Source Member RPGTNT in QCPYSRC</i> | 281 |
| 1. QUSEC — The API Error Data Structure | 281 |
| 2. OBJD0100 — Return Structure for QUSROBJD API | 281 |
| 3. Data Structures Used by QWCCVTDI API | 282 |
| 4. iconv()-Related Conversion APIs and Data Structures | 283 |
| 5. CEE4RAGE and CEE4RAGE2 Prototypes | 283 |
| 6. Regular Expression Base Data-types | 284 |
| 7. REGEX_T Data Structure Template | 284 |
| 8. REGMATCH_T Data Structure Template | 285 |
| 9. Regular Expression Named Constants | 285 |
| 10. Retrieve Environment Variable (RTVENVVAR) | 286 |