Institut für Betriebswirtschaftslehre und Wirtschaftsinformatik

Abteilung für Wirtschaftsinformatik

How to Develop a Native Library in C++ for ooRexx in a Nutshell

The 2015 International Rexx Symposium

Rony G. Flatscher

Wirtschaftsuniversität Wien ■ Welthandelsplatz 1 ■ A-1020 Wien

Agenda

- A nutshell example
- About argument types in native (C++) code
- About creating routines in native (C++) code
- About creating methods in native (C++) code
- Roundup and outlook

Nutshell Example, 1
C++ Code (extsimple.cpp)

```
#include <oorexxapi.h>
RexxRoutine0(int, NoArgRoutineReturn123)// return type & name
{
    return 123;                          // return value
}
RexxRoutineEntry orxtest_funcs[] = {
    REXX_TYPED_ROUTINE(NoArgRoutineReturn123 , NoArgRoutineReturn123 ),
    REXX_LAST_ROUTINE()                  // end marker
};

RexxMethod0(int, NoArgMethodReturn123) // return type & name
{
    return 123;                          // return value
}
RexxMethodEntry orxtest_meths[] = {
    REXX_METHOD(NoArgMethodReturn123 , NoArgMethodReturn123 ),
    REXX_LAST_METHOD()                   // end marker
};

RexxPackageEntry DemoExternalLibrary_package_entry = {
    STANDARD_PACKAGE_HEADER
    REXX_INTERPRETER_4_0_0,          // ooRexx version 4.0.0 or higher
    "ExternalDemo",                  // name of the package
    "1.0.0",                         // package information
    NULL, NULL,                      // no load and no unload function
    orxtest_funcs,                   // the exported routines
    orxtest_meths                    // the exported methods
};
OOREXX_GET_PACKAGE(DemoExternalLibrary);
```

Nutshell Example, 2
C++ Code, Windows Only (extsimple.def)

```
LIBRARY extsimple

EXPORTS
  RexxGetPackage
```

Nutshell Example, 3
ooRexx Code (testsimple.rex)

```
say "NoArgRoutineReturn123():" NoArgRoutineReturn123()

o=.test~new           -- create object
say "o~NoArgMethodReturn123 :" o~NoArgMethodReturn123

::routine NoArgRoutineReturn123 external "library extsimple NoArgRoutineReturn123"

::class test              -- define class and native methods for it
::method NoArgMethodReturn123   external "LIBRARY extsimple NoArgMethodReturn123"
```

Output:

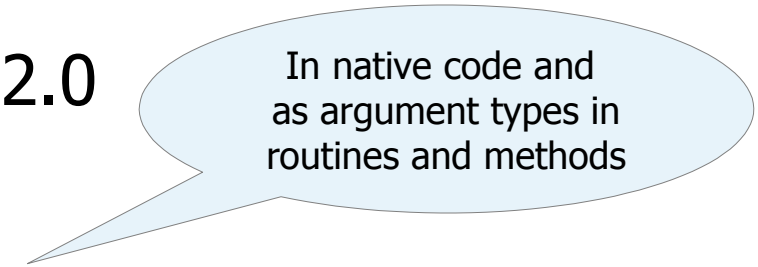
```
NoArgRoutineReturn123(): 123
o~NoArgMethodReturn123 : 123
```

Defining Library Routines and Library Methods, 1

- Official documentation
  - "rexx**pg**.pdf"
    - Part of the set of ooRexx documentation
  - Chapter 8
    - Cf. "8. Rexx C++ Application Programming Interfaces" (APIs)
      - Library routines
        - Cf. "8.12 Defining Library Routines"
      - Library methods
        - Cf. "8.13 Defining Library Methods"
- Test programs with ooRexx project on Sourceforge

Defining Library Routines and Library Methods, 2

- Native datatypes as of ooRexx 4.2.0
  - Cf. "8.2.1 Rexx Object Types"
    - RexxObjectPtr, RexxStringObject, RexxArrayObject, RexxClassObject, RexxMutableBufferObject, RexxStemObject
    - RexxBufferStringObject, RexxDirectoryObject, RexxSupplierObject, RexxPackageObject, RexxMethodObject, RexxRoutineObject, RexxPointerObject, RexxBufferObject
  - Cf. "8.2.2 Rexx Numeric types"
    - wholenumber_t, stringsize_t, logical_t, size_t, ssize_t, uintptr_t, intptr_t, int, int32_t, uint32_t, int64_t, uint64_t, int16_t, uint16_t, int8_t, uint8_t, float, double



Defining Library Routines and Library Methods, 3

- Helpful macros
  - Prefix "OPTIONAL_" for any argument
    - NULLOBJECT (NULL) or 0, if numeric datatype
    - Macros argumentExists(n) and argumentOmitted(n)
      - n is "1" based
  - Routines
    - RexxRoutineEntry structure
      - REXX_TYPED_ROUTINE(), REXX_CLASS_ROUTINE(), REXX_LAST_ROUTINE()

Defining Library Routines and Library Methods, 4

- Helpful macros (continued)
  - Methods
    - RexxMethodEntry structure
      - REXX_METHOD(), REXX_LAST_METHOD()
  - Additional, special types
    - Routines and methods
      - CSTRING, POINTER, POINTERSTRING
      - Not for return types
        - NAME, ARGLIST
    - Methods (not for return types)
      - OSELF, SUPER, SCOPE, CSELF

Defining Library Routines and Library Methods, 5

- Defined macros (cf. oorexxapi.h)
  - Routine functions
    - RexxRoutine0(returnType, name) ...
      RexxRoutine9(returnType, name, rt1, n1, ... rt9, n9)
  - Method functions
    - RexxMethod0(returnType, name) ...
      RexxMethod9(returnType, name, rt1, n1, ... rt9, n9)
- To define no return value
  - Return type RexxObjectPtr and return NULLOBJECT/NULL

Defining Library Routines Examples, 1
C++ Code (extfunc.cpp)

```
#include <oorexxapi.h>
#include <stdio.h>

// ------------------------------------------------------
RexxRoutine0(int,                       // return type
             NoArgRoutineReturn123)     // native routine name
{
    return 123;                         // return value
}

// ------------------------------------------------------
RexxRoutine0(RexxObjectPtr,             // return type a Rexx Object Pointer
             NoArgRoutineVoid)          // native routine name
{
    fprintf(stdout, "(from native code) \"NoArgRoutineVoid\"\n");
    return NULLOBJECT;                  // indicate no return value !
}

// ------------------------------------------------------
RexxRoutine1(int,                       // return type
             OneArgRoutineReturnArg,    // native routine name
             OPTIONAL_int, arg1)        // optional argument
{
    fprintf(stdout, "(from native code) \"OneArgRoutineReturnArg\": arg1=[%d]", arg1);
    fprintf(stdout, " argumentExists(1)=[%d], argumentOmitted(1)=[%d]\n",
                    argumentExists(1),      argumentOmitted(1));
    return arg1;                        // return value
}
```

Defining Library Routines Examples, 2
C++ Code (extfunc.cpp)

```
// ------------------------------------------------------
RexxRoutine1(RexxObjectPtr,          // return type a Rexx Object Pointer
             OneArgRoutineVoid,      // native routine name
             int, arg1)              // argument
{
    fprintf(stdout, "(from native code) \"OneArgRoutineVoid\": arg1=[%d]\n", arg1);
    return NULLOBJECT;               // indicate no return value !
}

// ------------------------------------------------------
RexxRoutine2(int,                    // return type
             TwoIntArgAdder,         // native routine name
             int, arg1,              // argument 1
             int, arg2)              // argument 2
{
    return arg1 + arg2;              // return value
}

// ------------------------------------------------------
RexxRoutine2(double,                 // return type
             TwoDoubleArgAdder,      // native routine name
             double, arg1,           // argument 1
             double, arg2)           // argument 2
{
    return arg1 + arg2;              // return value
}
```

Defining Library Routines Examples, 3
C++ Code (extfunc.cpp)

```
RexxRoutineEntry orxtest_funcs[] = {

    REXX_TYPED_ROUTINE(NoArgRoutineReturn123 , NoArgRoutineReturn123 ),
    REXX_TYPED_ROUTINE(NoArgRoutineVoid      , NoArgRoutineVoid      ),
    REXX_TYPED_ROUTINE(OneArgRoutineReturnArg, OneArgRoutineReturnArg),
    REXX_TYPED_ROUTINE(OneArgRoutineVoid     , OneArgRoutineVoid     ),
    REXX_TYPED_ROUTINE(TwoIntArgAdder        , TwoIntArgAdder        ),
    REXX_TYPED_ROUTINE(TwoDoubleArgAdder     , TwoDoubleArgAdder     ),
    REXX_LAST_ROUTINE()                    // end marker
};
```

```
RexxPackageEntry DemoExternalRoutines_package_entry = {
    STANDARD_PACKAGE_HEADER
    REXX_INTERPRETER_4_0_0,          // ooRexx version 4.0.0 or higher
    "ExternalRoutineDemo",           // name of the package
    "1.0.0",                         // package information
    NULL,                            // no load function
    NULL,                            // no unload function
    orxtest_funcs,                   // the exported routines
    NULL                             // the exported methods
};

// package loading stub.
OOREXX_GET_PACKAGE(DemoExternalRoutines);
```

Defining Library Routines Examples, 4
C++ Code, Windows Only (extfunc.def)

```
LIBRARY extfunc

EXPORTS
  RexxGetPackage
```

Using Library Routines Examples, 1
C++ Code (testextfunc.cpp)

```
say "NoArgRoutineReturn123()     :" pp(NoArgRoutineReturn123())
say
say "calling a routine that returns nothing:"
call NoArgRoutineVoid
say
say "OneArgRoutineReturnArg(1)   :" pp(OneArgRoutineReturnArg(1))
say "OneArgRoutineReturnArg( )   :" pp(OneArgRoutineReturnArg())
say
say "calling another routine that returns nothing:"
call OneArgRoutineVoid 2
say
say "TwoIntArgAdder(4, 5)        :" pp(TwoIntArgAdder(4, 5))
say "TwoDoubleArgAdder(6   , 8  ):" pp(TwoDoubleArgAdder(6   , 8  ))
say "TwoDoubleArgAdder(6.7, 8.9):" pp(TwoDoubleArgAdder(6.7, 8.9))
say "---"

say "Invoking an external routine with wrong arguments:"
say "TwoIntArgAdder(9.8, 7.6)    :" pp(TwoIntArgAdder(9.8, 7.6))

::routine pp   -- return string value enclosed in square brackets
  return "["arg(1)"]"
```

```
::routine NoArgRoutineReturn123  external "library extfunc NoArgRoutineReturn123"
::routine NoArgRoutineVoid       external "library extfunc NoArgRoutineVoid"
::routine OneArgRoutineReturnArg external "library extfunc OneArgRoutineReturnArg"
::routine OneArgRoutineVoid      external "library extfunc OneArgRoutineVoid"
::routine TwoDoubleArgAdder      external "library extfunc TwoDoubleArgAdder"
```

Using Library Routines Examples, 2
C++ Code (testextfunc2.cpp)

```
say "NoArgRoutineReturn123()     :" pp(NoArgRoutineReturn123())
say
say "calling a routine that returns nothing:"
call NoArgRoutineVoid
say
say "OneArgRoutineReturnArg(1)   :" pp(OneArgRoutineReturnArg(1))
say "OneArgRoutineReturnArg( )   :" pp(OneArgRoutineReturnArg())
say
say "calling another routine that returns nothing:"
call OneArgRoutineVoid 2
say
say "TwoIntArgAdder(4, 5)        :" pp(TwoIntArgAdder(4, 5))
say "TwoDoubleArgAdder(6   , 8  ):" pp(TwoDoubleArgAdder(6   , 8  ))
say "TwoDoubleArgAdder(6.7, 8.9):" pp(TwoDoubleArgAdder(6.7, 8.9))
say "---"

say "Invoking an external routine with wrong arguments:"
say "TwoIntArgAdder(9.8, 7.6)    :" pp(TwoIntArgAdder(9.8, 7.6))

::requires "extfunc" library

::routine pp   -- return string value enclosed in square brackets
  return "["arg(1)"]"
```

Running Library Routines Examples, 2
C++ Code (testextfunc.cpp/textextfunc2.cpp)

Output:

```
E:\\201504-RexxLA\external\code>testextroutine.rex
NoArgRoutineReturn123()     : [123]

calling a routine that returns nothing:
(from native code) "NoArgRoutineVoid"

(from native code) "OneArgRoutineReturnArg": arg1=[1] argumentExists(1)=[1],
argumentOmitted(1)=[0]
OneArgRoutineReturnArg(1)   : [1]
(from native code) "OneArgRoutineReturnArg": arg1=[0] argumentExists(1)=[0],
argumentOmitted(1)=[1]
OneArgRoutineReturnArg( )   : [0]

calling another routine that returns nothing:
(from native code) "OneArgRoutineVoid": arg1=[2]

TwoIntArgAdder(4, 5)        : [9]
TwoDoubleArgAdder(6   , 8   ): [14]
TwoDoubleArgAdder(6.7, 8.9): [15.6]
---
Invoking an external routine with wrong arguments:
       *-* Compiled routine TWOINTARGADDER
    18 *-* say "TwoIntArgAdder(9.8, 7.6)    :" pp(TwoIntArgAdder(9.8, 7.6))
Error 88 running E:\DropBox\Dropbox\Vortraege\201504-
RexxLA\external\code\testextroutine.rex line 18:  Invalid argument
Error 88.907:  Argument 1 must be in the range -2147483648 to 2147483647; found "9.8"
```

Defining Library Methods Examples, 1
C++ Code (extmeth.cpp)

```
#include <oorexxapi.h>
#include <stdio.h>

// ------------------------------------------------------
RexxMethod0(int,                        // return type
            NoArgMethodReturn123)       // native method name
{
    return 123;                         // return value
}

// ------------------------------------------------------
RexxMethod0(RexxObjectPtr,              // return type a Rexx Object Pointer
            NoArgMethodVoid)            // native method name
{
    fprintf(stdout, "(from native code) \"NoArgMethodVoid\"\n");
    return NULLOBJECT;                  // indicate no return value !
}

// ------------------------------------------------------
RexxMethod1(int,                        // return type
            OneArgMethodReturnArg,      // native method name
            OPTIONAL_int, arg1)                 // argument
{
    fprintf(stdout, "(from native code) \"OneArgMethodReturnArg\": arg1=[%d]", arg1);
    fprintf(stdout, " argumentExists(1)=[%d], argumentOmitted(1)=[%d]\n",
                    argumentExists(1),      argumentOmitted(1));
    return arg1;                        // return value
}
```

Defining Library Methods Examples, 2
C++ Code (extmeth.cpp)

```
// ------------------------------------------------------
RexxMethod1(RexxObjectPtr,           // return type a Rexx Object Pointer
            OneArgMethodVoid,        // native method name
            int, arg1)               // argument
{
    fprintf(stdout, "(from native code) \"OneArgMethodVoid\": arg1=[%d]\n", arg1);
    return NULLOBJECT;               // indicate no return value !
}

// ------------------------------------------------------
RexxMethod2(int,                     // return type
            TwoIntArgAdder,          // native method name
            int, arg1,               // argument 1
            int, arg2)               // argument 2
{
    return arg1 + arg2;              // return value
}

// ------------------------------------------------------
RexxMethod2(double,                  // return type
            TwoDoubleArgAdder,       // native method name
            double, arg1,            // argument 1
            double, arg2)            // argument 2
{
    return arg1 + arg2;              // return value
}
```

Defining Library Methods Examples, 3
C++ Code (extmeth.cpp)

```
RexxMethod2(RexxObjectPtr,           // return type a Rexx Object Pointer
            OneArgInvokeOrxMethod,   // native method name
            OSELF, self,             // pseudo-argument: allow access to object instance
            int, arg1)               // argument from Rexx program!
{
    fprintf(stdout, "(from native code) \"OneArgInvokeOrxMethod\": arg1=[%d]\n", arg1);
    fprintf(stdout, "(from native code) invoking method \"hello\" in this very same object:\n", arg1);
    context->SendMessage0(self,"HELLO");
    return NULLOBJECT;               // indicate no return value !
}
```

```
RexxMethodEntry orxtest_meths[] = {

    REXX_METHOD(NoArgMethodReturn123 , NoArgMethodReturn123 ),
    REXX_METHOD(NoArgMethodVoid      , NoArgMethodVoid      ),
    REXX_METHOD(OneArgMethodReturnArg, OneArgMethodReturnArg),
    REXX_METHOD(OneArgMethodVoid     , OneArgMethodVoid     ),
    REXX_METHOD(TwoIntArgAdder       , TwoIntArgAdder       ),
    REXX_METHOD(TwoDoubleArgAdder    , TwoDoubleArgAdder    ),
    REXX_METHOD(OneArgInvokeOrxMethod, OneArgInvokeOrxMethod),
    REXX_LAST_METHOD()               // end marker
};
```

Defining Library Methods Examples, 4
C++ Code (extmeth.cpp)

```
RexxPackageEntry DemoExternalMethods_package_entry = {
    STANDARD_PACKAGE_HEADER
    REXX_INTERPRETER_4_0_0,          // ooRexx version 4.0.0 or higher
    "ExternalMethodDemo",            // name of the package
    "1.0.0",                         // package information
    NULL,                            // no load function
    NULL,                            // no unload function
    NULL,                            // the exported routines
    orxtest_meths                    // the exported methods
};
```

```
// package loading stub.
OOREXX_GET_PACKAGE(DemoExternalMethods);
```

Defining Library Methods Examples, 5
C++ Code, Windows Only (extmeth.def)

```
LIBRARY extmeth

EXPORTS
  RexxGetPackage
```

Using Library Methods Examples, 1
C++ Code (testextmethod.rex)

```
o=.test~new          -- create object
o~hello
say "---"
say "o~NoArgMethodReturn123()     :" pp(o~NoArgMethodReturn123)
say
say "invoking a method that returns nothing:"
o~NoArgMethodVoid
say
say "o~OneArgMethodReturnArg(1)   :" pp(o~OneArgMethodReturnArg(1))
say "o~OneArgMethodReturnArg( )   :" pp(o~OneArgMethodReturnArg( ))
say
say "invoking another method that returns nothing:"
o~OneArgMethodVoid(2)
say
say "o~TwoIntArgAdder(4, 5)         :" pp(o~TwoIntArgAdder(4, 5))
say "o~TwoDoubleArgAdder(6   , 8  ):" pp(o~TwoDoubleArgAdder(6   , 8  ))
say "o~TwoDoubleArgAdder(6.7, 8.9):" pp(o~TwoDoubleArgAdder(6.7, 8.9))
say "---"
say "invoking a method that natively will send 'hello' to its object:"
o~OneArgInvokeOrxMethod(-123)
say "---"

say "Invoking an external method with wrong arguments:"
say "o~TwoIntArgAdder(9.8, 7.6)    :" pp(o~TwoIntArgAdder(9.8, 7.6))

… directives on next slide ...
```

Using Library Methods Examples, 2
C++ Code (testextmethod.rex)

```
... directives: ….

::routine pp   -- return string value enclosed in square brackets
  return "["arg(1)"]"

::class test             -- define class and native methods for it
::method NoArgMethodReturn123  external "LIBRARY extmeth NoArgMethodReturn123 "
::method NoArgMethodVoid       external "LIBRARY extmeth NoArgMethodVoid      "
::method OneArgMethodReturnArg external "LIBRARY extmeth OneArgMethodReturnArg"
::method OneArgMethodVoid      external "LIBRARY extmeth OneArgMethodVoid     "
::method TwoIntArgAdder        external "LIBRARY extmeth TwoIntArgAdder       "
::method TwoDoubleArgAdder     external "LIBRARY extmeth TwoDoubleArgAdder    "
::method OneArgInvokeOrxMethod external "LIBRARY extmeth OneArgInvokeOrxMethod"

::method hello
  say "hello from" pp(self) "self~identityHash:" pp(self~identityHash)
```

Running Library Methods Example, 2
Output (testextmethod.rex), 1

Output:

```
E:\201504-RexxLA\external\code>testextmethod.rex
hello from [a TEST] self~identityHash: [266199628]
---
o~NoArgMethodReturn123()    : [123]

invoking a method that returns nothing:
(from native code) "NoArgMethodVoid"

(from native code) "OneArgMethodReturnArg": arg1=[1] argumentExists(1)=[1],
argumentOmitted(1)=[0]
o~OneArgMethodReturnArg(1)  : [1]
(from native code) "OneArgMethodReturnArg": arg1=[0] argumentExists(1)=[0],
argumentOmitted(1)=[1]
o~OneArgMethodReturnArg( )  : [0]

invoking another method that returns nothing:
(from native code) "OneArgMethodVoid": arg1=[2]

o~TwoIntArgAdder(4, 5)       : [9]
o~TwoDoubleArgAdder(6  , 8  ): [14]
o~TwoDoubleArgAdder(6.7, 8.9): [15.6]
---
… contiued on next slide …
```

Running Library Methods Example, 3 Output (testextmethod.rex), 2

```
… continued from previous slide …

---
invoking a method that natively will send 'hello' to its object:
(from native code) "OneArgInvokeOrxMethod": arg1=[-123]
(from native code) invoking method "hello" in this very same object:
hello from [a TEST] self~identityHash: [266199628]
---
Invoking an external method with wrong arguments:
       *-* Compiled method TWOINTARGADDER with scope "TEST"
    24 *-* say "o~TwoIntArgAdder(9.8, 7.6)   :" pp(o~TwoIntArgAdder(9.8, 7.6))
Error 88 running E:\DropBox\Dropbox\Vortraege\201504-
RexxLA\external\code\testextmethod.rex:  Invalid argument
Error 88.907:  Argument 1 must be in the range -2147483648 to 2147483647; found "9.8"
```

Roundup and Outlook

- Roundup
  - Powerful ooRexx C++ APIs !
  - Easy to create external („native“) routines and methods !
  - No excuse whatsoever to not create your own external libraries for ooRexx!
    ;-)