# RPG Programming

## Tips and Tricks


**Liam Allan**

# Agenda

- Not just tips and tricks
- Also ‘do not do this’ content
- Maybe some IDE tricks here and there
- Very opinionated

# Wrong structs are valid in parameters

- In general, RPG thinks that data structures are also character strings
- Allowed to pass any data structure or any string to a parameter defined with LIKEDS

```
dcl-pr check_order ind;
   order_info likeds(order_t);
   customer_info likeds(customer_t) const;
end-pr;

ok = check_order (customer : order);      // Bug!
ok = check_order (order : customer.name); // Bug!
```

# Use OPTIONS(*EXACT) on parameters

- The compiler will only allow you to pass a data structure defined with the same LIKEDS
- The compile fails because the parameters don't meet the OPTIONS(*EXACT) requirements

```
dcl-pr check_order ind;
   order_info likeds(order_t) options(*exact);
   customer_info likeds(customer_t) const options(*exact);
end-pr;

ok = check_order (customer : order);       // Compile error
ok = check_order (order : customer.name);  // Compile error
```

# Mixed cased procedures

- By default, RPGLE procedures are uppercase
- Call stack will reflect casing
- Mixed is prettier and simple to use

```
5  QUOCMD     QSYS                    /03B3
    PGMSTK     BMORRIS                         _QRNP_PEP_PGMSTK
    PGMSTK     BMORRIS       6                 HANDLEORDER
    PGMSTK     BMORRIS       10                CHECKCUSTSTATUS
    PGMSTK     BMORRIS       14                GETCUSTID
```

```
5  QUOCMD     QSYS                    /03B3
    PGMSTK     BMORRIS                         _QRNP_PEP_PGMSTK
    PGMSTK     BMORRIS       6                 handleOrder
    PGMSTK     BMORRIS       10                checkCustStatus
    PGMSTK     BMORRIS       14                getCustId
```

# Mixed case procedures

If you add or change EXTPROC:

- Recompile everything that uses the prototype
- Fix your CL to have the mixed-case name:
  `CALLPRC 'handleOrder'`
- Fix your binder source to have the mixed-case
  `EXPORT(“handleOrder”)`

```
dcl-proc handleOrder;
   dcl-pi *n extproc(*dclcase) end-pi;
   return;
end-proc;
```

# Always with INLR

- So used to using *INLR
- Tells the cycle to end
- Can also use RETURN

```
**free

dcl-s thetext char(52);

thetext = 'Hello world';

dsply thetext;

*inlr = *on;
```

# How about.. no more LR?

- Header to define what procedure is the entry point
- Allows for cleaning and grouped code
- Program PI defined inside of procedure
- Removes the need to set LR

```
**free

ctl-opt main(sayHello);

dcl-proc sayHello;
  dcl-s thetext char(52);

  thetext = 'Hello world';

  dsply thetext;
end-proc;
```

# Trims are ugly and slow

- If you have a lot of %TRIM in your code
  - Consider using varying-length strings instead
- %TRIM starts at the end of the field and works backward until it finds a non-blank character
- Varying-length fields know how much data they have, so it's not usually necessary to trim the data

```
**free
dcl-pi *n;
  lib char(10) const;
  file char(10) const;
end-pi;

dcl-s cmd char(100);

cmd = 'DSPPFM ' + %trim(lib) + '/' + %trim(file)
    + ' OUTPUT(*PRINT)';
callp(e) QCMDEXC (cmd : %len(%trim(cmd)));

if %error();
  report ('File ' + %trim(file) + 'does not exist '
         + 'in library ' + %trim(lib) + '.');
endif;
```

# Trim minimally

- Minimal trims
- VARCHAR type

```
**free
dcl-pi *n;
  libParm char(10) const;
  fileParm char(10) const;
end-pi;

dcl-s lib varchar(10);
dcl-s file varchar(10);
dcl-s cmd varchar(100);

lib = %trim(libParm);
file = %trim(fileParm);

cmd = 'DSPPFM ' + lib + '/' + file + ' OUTPUT(*PRINT)';
callp(e) QCMDEXC (cmd : %len(cmd));

if %error();
  report ('File ' + file + 'does not exist '
         + 'in library ' + lib + '.');
endif;
```

# No manual trim at all?

- The hard way - always remember to code %TRIM

```
dcl-pr getFileInfo;
  file varchar(101) const;
end-pr;
getFileInfo (%trim(filename));
```

- The easy way - let OPTIONS(*TRIM) handle trimming

```
dcl-pr getFileInfo;
  file varchar(101) const options(*trim);
end-pr;
getFileInfo (filename);
```

# Help out the newbies, be explicit (QUALIFIED)

- Newcomers need help
- Implicit references are annoying

```
read ordRec;
dow not %eof(ord92);
  ok = checkInventory (city : item_id : quantity);
  read ordRec;
enddo;
```

```
read ord92.ordRec order; // File DS's are qualified
dow not %eof(ord92);
  ok = checkInventory (
    cust.city :
    order.item_id :
    order.quantity
  );
  read ord92.ordRec order;
enddo;
```

# Qualify it all

- Qualify from record formats
- Qualify entire file definitions

```
dcl-f orders;
dcl-ds orderDs likerec(orderRec);

read orderRec orderDs; // read into the DS
```

```
dcl-f orders qualified;
dcl-ds orderDs likerec(orderRec);

read orders.orderRec orderDs; // read into the DS
```

# Lovely, pretty names

- Long name
  - SQL name
- Short name
  - System name

```
CREATE OR REPLACE TABLE DEPARTMENT (
  department_id FOR DEPTNO CHAR(3) NOT NULL,
  deptartment_name FOR DEPTNAME VARCHAR(36) NOT NULL,
  manager_number FOR MGRNO CHAR(6) NOT NULL,
  parent_department FOR ADMRDEPT CHAR(3) NOT NULL,
  LOCATION CHAR(16) NOT NULL,
  PRIMARY KEY (DEPTNO)
);
```

# Lovely, pretty names… in RPGLE

- Without ALIAS, ugly short names
  - Harder for newbies too!
- Using ALIAS
  - Better readability
  - Easier for your pals

```
**free
dcl-f orders;
// ...
if ordqty > 0;
  placeOrder(ordId : cstId : ordqty
             : splcty : cstcty); // "splcty" ???
endif;
```

```
**free
dcl-f orders alias;
// ...
if order_quantity > 0;
  placeOrder(order_id : customer_id : order_quantity
             : supplier_city : customer_city);
endif;
```

# New data-structure definition

```
dcl-ds emp_t qualified template;
  name varchar(25);
  salary packed(7 : 2);
  is_manager ind;
end-ds;

dcl-ds dept qualified;
  num_emps int(10);
  emps likeds(emp_t) dim(30);
end-ds;
```

```
dcl-ds dept qualified;
  num_emps int(10);
  dcl-ds emps dim(30);
    name varchar(25);
    salary packed(7 : 2);
    is_manager ind;
  end-ds;
end-ds;
```

# Exit handler

- Put the cleanup tasks in the ON-EXIT section of the procedure.
- The ON-EXIT section is always run, no matter how the procedure ends.

```
**free
dcl-proc myproc;
  // …

  p = %alloc(1000);
  // …

  if not %found;
    return;
  endif;
  // …

on-exit;
  dealloc p;
end-proc;
```

# No local prototype

- Prototype not needed for local procedures
- Only needed for EXTPGM or EXTPROC

```
dcl-pr PRTTXT extpgm;
  output char(52);
end-pr;

// dcl-pr writeOut;
//   output char(52);
// end-pr;

dcl-proc writeOut;
  dcl-pi *n;
    output char(52);
  end-pi;

  PRTTXT(output);
end-proc;
```

# Love your constants

- Re-used strings are ugly
- Make it much easier to refactor by using constants

```
dcl-pi *n;
  textIn chat(32);
end-pi;

dcl-s mytext char(52);

mytext = textIn;

if (mytext = *blank);
  mytext = 'HI2';
endif;

select;
  when (mytext = 'HI1');
    dsply 'Hello world';
  when (mytext = 'HI2');
    dsply 'Goodbye world';
  other;
    dsply 'Hello world';
endsl;
```

# Love your constants

- Re-used strings are ugly
- Make it much easier to refactor by using constants
- A good IDE makes constants easy

```
17   select;
18     when (my   HI1_TEXT : 'HELLO WORLD'
19       dsply HI1_TEXT;
20     when (mytext = HI2);
21       dsply HI2_TEXT;
22     other;
23       dsply HI1_TEXT;
24   endsl;
```

```
// IDs
dcl-c HI1 'HI1';
dcl-c HI2 'HI2';

// Messages
dcl-c HI1_TEXT 'Hello world';
dcl-c HI2_TEXT 'Goodbye world';

//...

if (mytext = *blank);
  mytext = HI2;
endif;

select;
  when (mytext = HI1);
    dsply HI1_TEXT;
  when (mytext = HI2);
    dsply HI2_TEXT;
  other;
    dsply HI1_TEXT;
endsl;
```

# Take care of your structs

- RPG considers a data structure to be also a character string.
- You can assign one data structure to another using EVAL.
- This is fine as long as they
  - have identical subfields, AND
  - don't have any null-capable subfields

```
dcl-ds ds1;
  subfa char(60);
  subfb int(10);
end-ds;

dcl-ds ds2;
  subfa char(30);
  subfb int(10);
end-ds;

eval ds1 = ds2;
```

# Take care of your structs

- Rather than using EVAL, use EVAL-CORR ("corresponding").
- EVAL-CORR assigns subfield by subfield.
- Subfields that have the same name and compatible data types are assigned.
- Null indicators are also assigned for null-capable subfields.
- Other subfields are ignored.

```
dcl-ds ds1;
  subfa char(60);
  subfb int(10);
  subfc packed(10:5);
end-ds;

dcl-ds ds2;
  subfa char(30);
  subfb int(10);
end-ds;

eval-corr ds1 = ds2;
```

# Talk to the world

- New, faster, UDTFs for sending HTTP requests
- No more Java
- Easy to use

```
**free

Dcl-Ds request Qualified;
  URL    Varchar(64);
  Header Varchar(1024);
  Body   Varchar(2048);
End-Ds;

request.Header
  =  '{'
  + '"header": "Content-Type,application/json",'
  + '}';

request.Body = json_AsText(pBody);

EXEC SQL
  SET :Response =
  QSYS2.HTTP_POST(
    :request.URL,
    :request.Body,
    :request.Header
  );
```

# Talk to the world in many ways

HTTP_DELETE
HTTP_DELETE_VERBOSE

HTTP_GET
HTTP_GET_VERBOSE

HTTP_PUT
HTTP_PUT_VERBOSE

HTTP_POST
HTTP_POST_VERBOSE

# Use any UDTF or scalar function

- Embedded SQL lets you run any SQL
- Including grabbing result sets
- So many UDTFs and scalar functions

```
**free

Dcl-S VarcharField Varchar(256);

VarcharField = 'Hello world!';

EXEC SQL SET :VarcharField = QSYS2.BASE64_ENCODE(:VarcharField);
EXEC SQL SET :VarcharField = QSYS2.BASE64_DECODE(:VarcharField);
```

# No execute immediate, ever

- Terrible readability
- Broken if a single quote is actually used
- Open to SQL injection attack
  - Esp. with LIKE

```
Dcl-S personName Varchar(60);
Dcl-S deleteStatement Varchar(200);

deleteStatement
  = 'delete from users where '
  + 'user_fullname = ''' + personName + '''';

EXEC SQL EXECUTE IMMEDIATE :deleteStatement;
```

# Be more explicit (no more SELECT *)

- Improved readability
- Easier for developers
  - Esp. new ones
- Removes possible bugs

```
// Static
Dcl-Ds users extfile('users');

// Not static
EXEC SQL
  SELECT * INTO :users
  FROM users
  WHERE ID = 1;
```

# Don’t write ugly code

- Don’t write joins in your RPGLE
- Can easily get complex
  - This is basic

```
dcl-s empno char(6);
dcl-s firstName char(20);
dcl-s workdept char(8);
dcl-s deptName char(30);

EXEC SQL
  select
    e.empno, e.firstnme, e.workdept, d.deptname
  into
    :empno, :firstName, :workdept, :deptName
  from sample.employee as e
  left join sample.department as d
    on e.workdept = d.deptno
  where empno = '00100'
```

# Use a view

```
create or replace view empDepartments as
  select
    e.empno, e.firstnme, e.workdept, d.deptname
  from sample.employee as e
  left join sample.department as d
    on e.workdept = d.deptno;
```

```
dcl-s empno char(6);
dcl-s firstName char(20);
dcl-s workdept char(8);
dcl-s deptName char(30);

EXEC SQL
  select
    e.empno, e.firstnme, e.workdept, d.deptname
  into
    :empno, :firstName, :workdept, :deptName
  from empDepartments
  where empno = '00100';
```

# Any more?

# Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

# Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, Db2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, Db2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, , GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: http://www.**ibm.com**/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
AltiVec is a trademark of Freescale Semiconductor, Inc.
AMD Opteron is a trademark of Advanced Micro Devices, Inc.
InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.
Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.
NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.
SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).
The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.
TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).
UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.