

the awesomeness of
SERVICE PROGRAMS

and the truth about binder source

Patrick Behr

Death and taxes

...and program logic needing to be changed

How much we got?

```
dcl-f  INVBAL  keyed;
```

```
CHAIN (loc : item)  INVBAL;  
weGot = ONHAND;
```

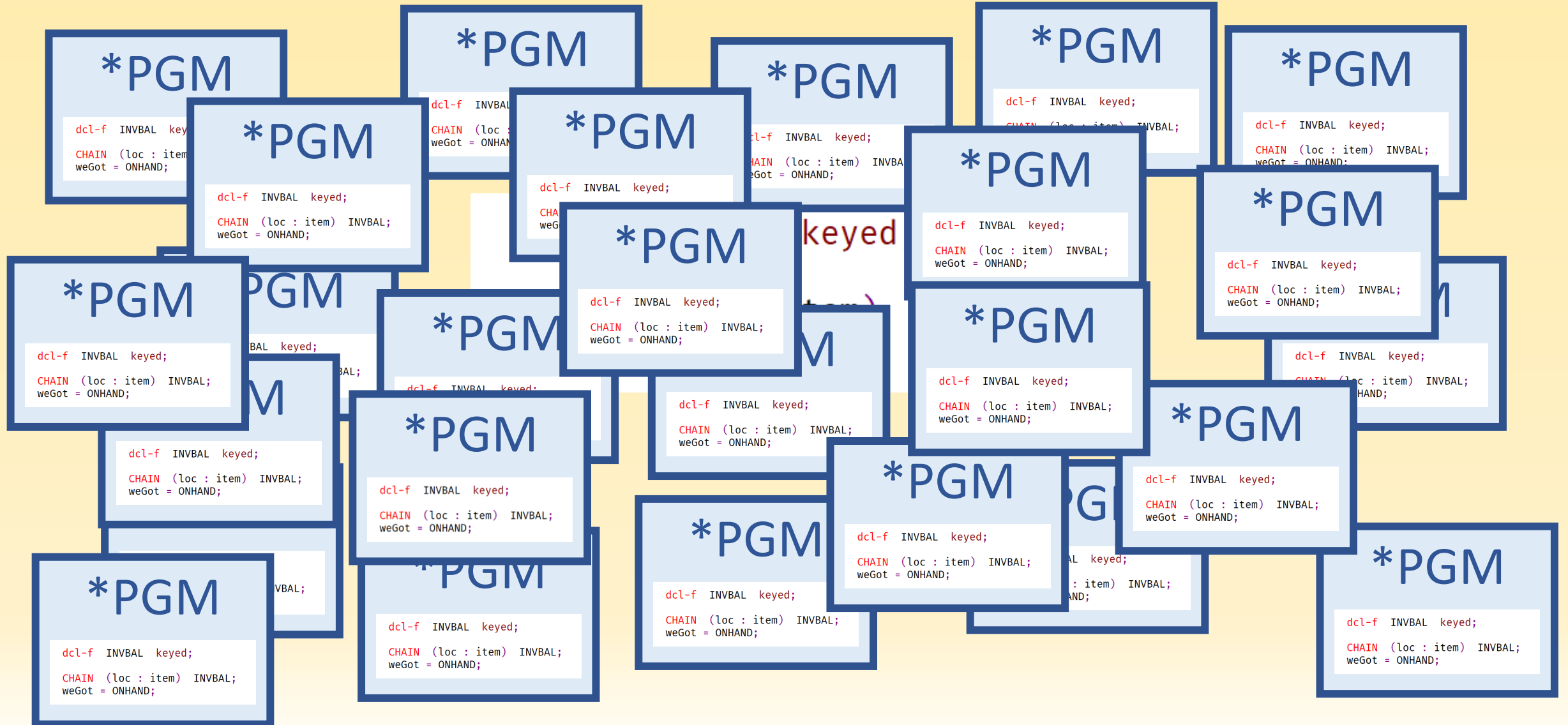
How much we got?

```
dcl-f  INVBAL      keyed;
dcl-f  REPLACEITM keyed;

CHAIN (loc : item) INVBAL;
weGot = ONHAND;

CHAIN (item) REPLACEITM;
if %found(REPLACEITM);
    CHAIN (loc : rplitem) INVBAL;
    weGot += ONHAND;
endif;
```

How much we got?



#^&* @!



Alternatives

- OPM

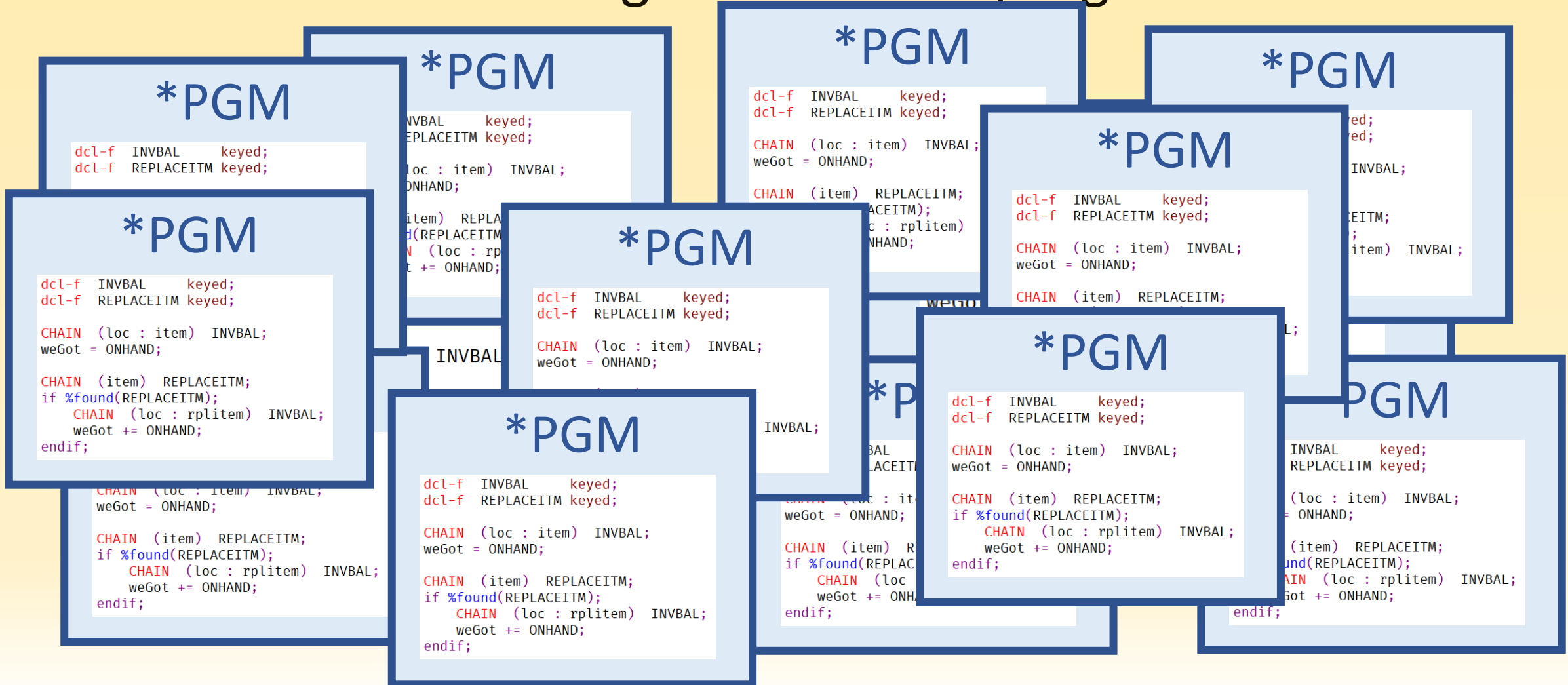
- Hard-code the on-hand logic into all the programs
- New program for on-hand logic + dynamic program calls

- ILE

- New module for on-hand logic + bind by copy
- New service program for on-hand logic + bind by reference

Alternatives - OPM

- Hard-code the new logic into all the programs



Alternatives - OPM

- Hard-code the new logic into all the programs
 - You have to crack them all open anyway ... and it's easy
 - Add more logic
 - Get it right every time, in every program
 - Do it all again next time for the next change



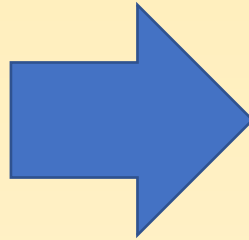
Alternatives - OPM

- Create a new *PGM for on-hand calcs

*PGM

```
dcl-pr GetOnHand extpgm('GETONHAND');
  in_location    packed(5) const;
  in_item       packed(7) const;
  out_onhand    packed(11);
end-pr;

GetOnHand( loc : item : weGot );
```



GETONHAND *PGM

```
dcl-f INVBAL keyed;
dcl-f REPLACEITM keyed;

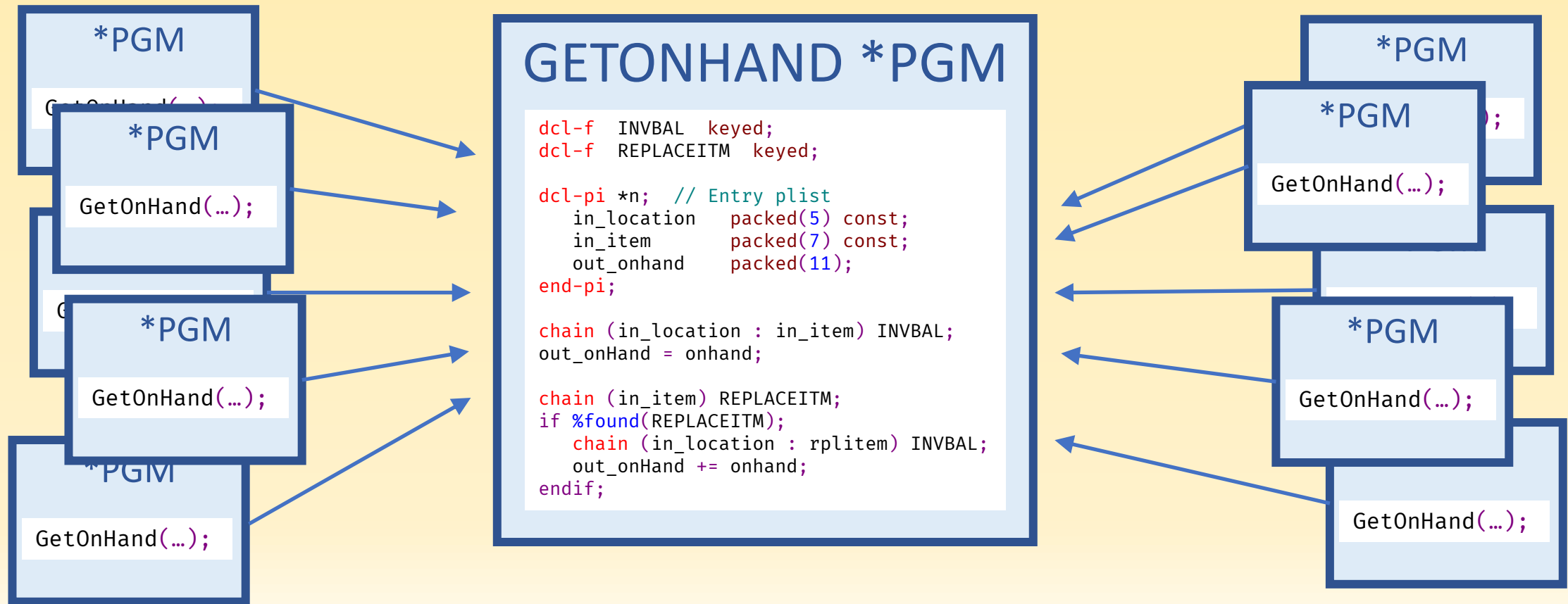
dcl-pi *n; // Entry plist
  in_location    packed(5) const;
  in_item       packed(7) const;
  out_onhand    packed(11);
end-pi;

chain (in_location : in_item) INVBAL;
out_onHand = onhand;

chain (in_item) REPLACEITM;
if %found(REPLACEITM);
  chain (in_location : rplitem) INVBAL;
  out_onHand += onhand;
endif;
```

Alternatives - OPM

- Create a new *PGM for on-hand calcs



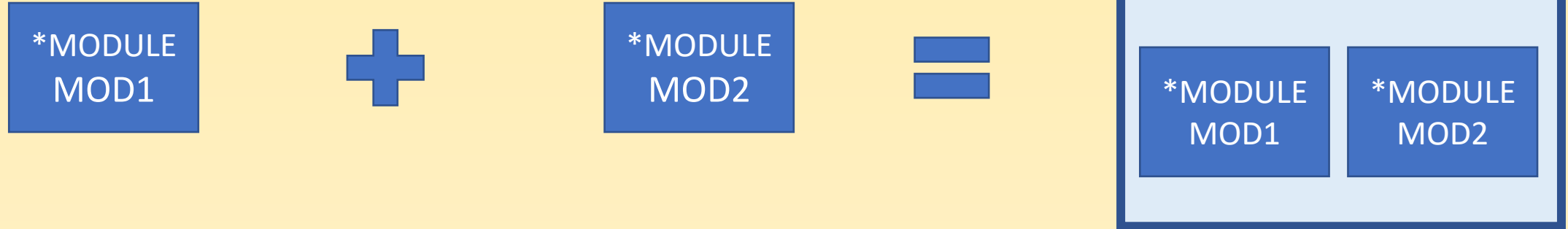
Alternatives - OPM

- Create a new program for on-hand calcs, and CALL that
 - You have to crack them all open anyway...and this is easy too
 - A modular approach
 - Program must have 10-character name
 - A separate program object for each shared function
 - Overhead of dynamic program calls
 - Can return a parameter, but not a value



Alternatives - ILE

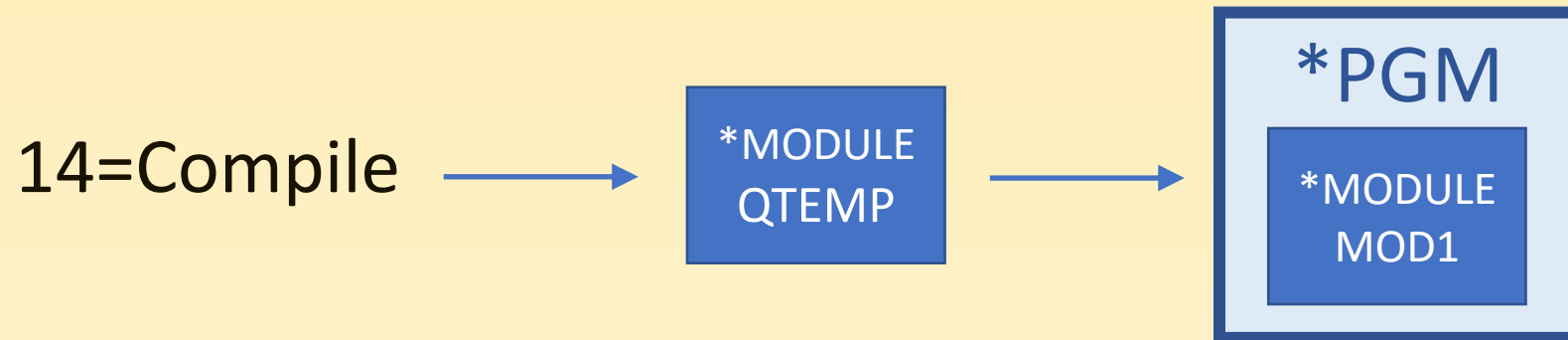
- Bind by copy



<side note>

- Bind by copy

*MODULE = *PGM



Create Bound RPG Program (CRTBNDRPG)

Alternatives - ILE

- Bind by copy

*MODULE + *MODULE = *PGM

Existing Source

```
dcl-pr GetOnHand extpgm('GETONHAND');  
in_location  packed(5) const;  
in_item      packed(7) const;  
out_onhand   packed(11);  
end-pr;  
  
GetOnHand( loc : item : weGot );
```



GetOnHand Source

```
chain (in_location : in_item) INVBAL;  
out_onHand = onhand;  
  
chain (in_item) REPLACEITM;  
if %found(REPLACEITM);  
  chain (in_location : rplitem) INVBAL;  
  out_onHand += onhand;  
endif;
```

CRTPGM

CRTRPGMOD

*MODULE
MOD1



CRTRPGMOD

*MODULE
MOD2



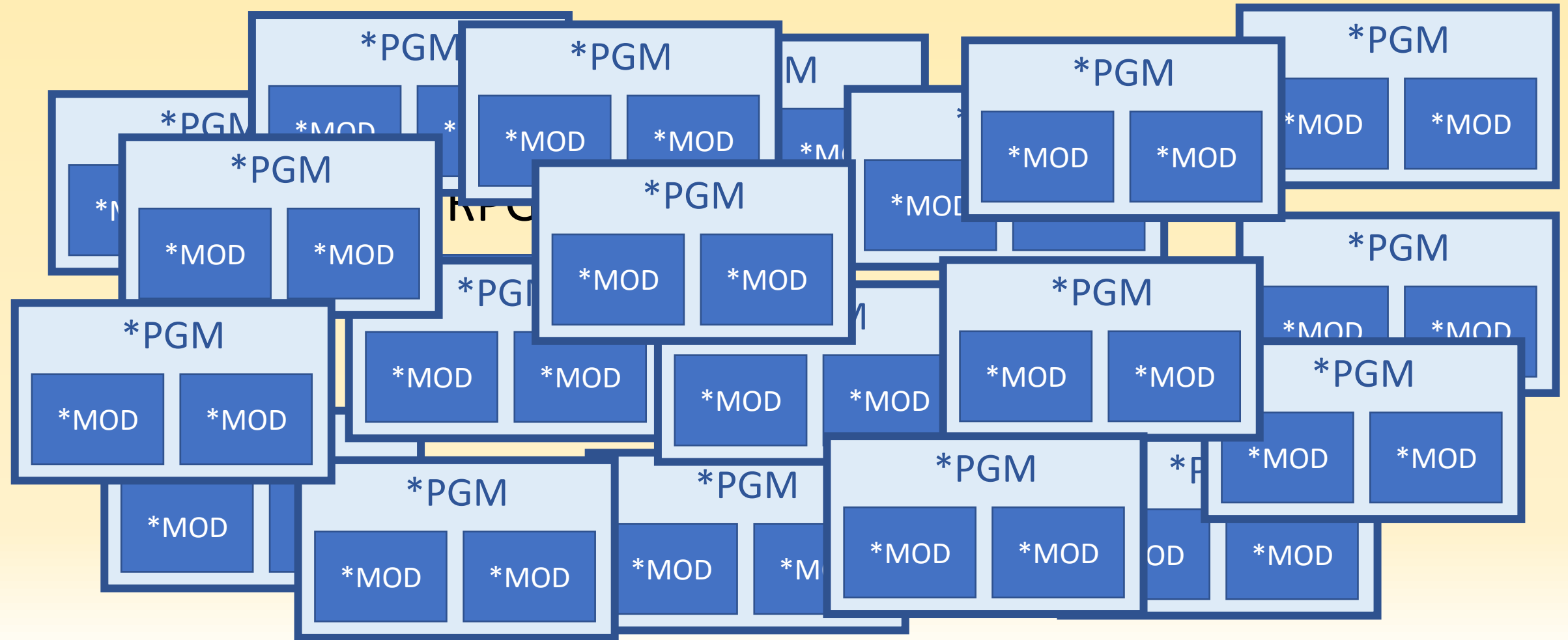
*PGM

*MODULE
MOD1

*MODULE
MOD2

Alternatives - ILE

- Create a new module for on-hand calcs, and BIND by copy



Alternatives - ILE

- Create a new module for on-hand calcs, and BIND by copy
 - It would be very fast
 - It would be modularized
- Keep all your *MODULEs
- Rebind all your programs for the next change

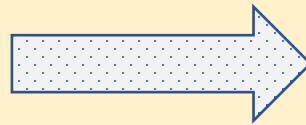
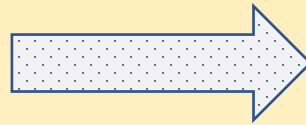
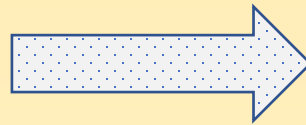


Alternatives - ILE

- Bind by reference
*PGM + *SRVPGM

*PGM

```
dcl-f SOMEFILE usage(*UPDATE) keyed;  
  
dcl-s someVar char(10) dtaara('DTAARA');  
  
dcl-pr GetOnHand;  
  parameters...  
end-pr;  
  
onhand = GetOnhand('LOC' : 'ITEM');
```

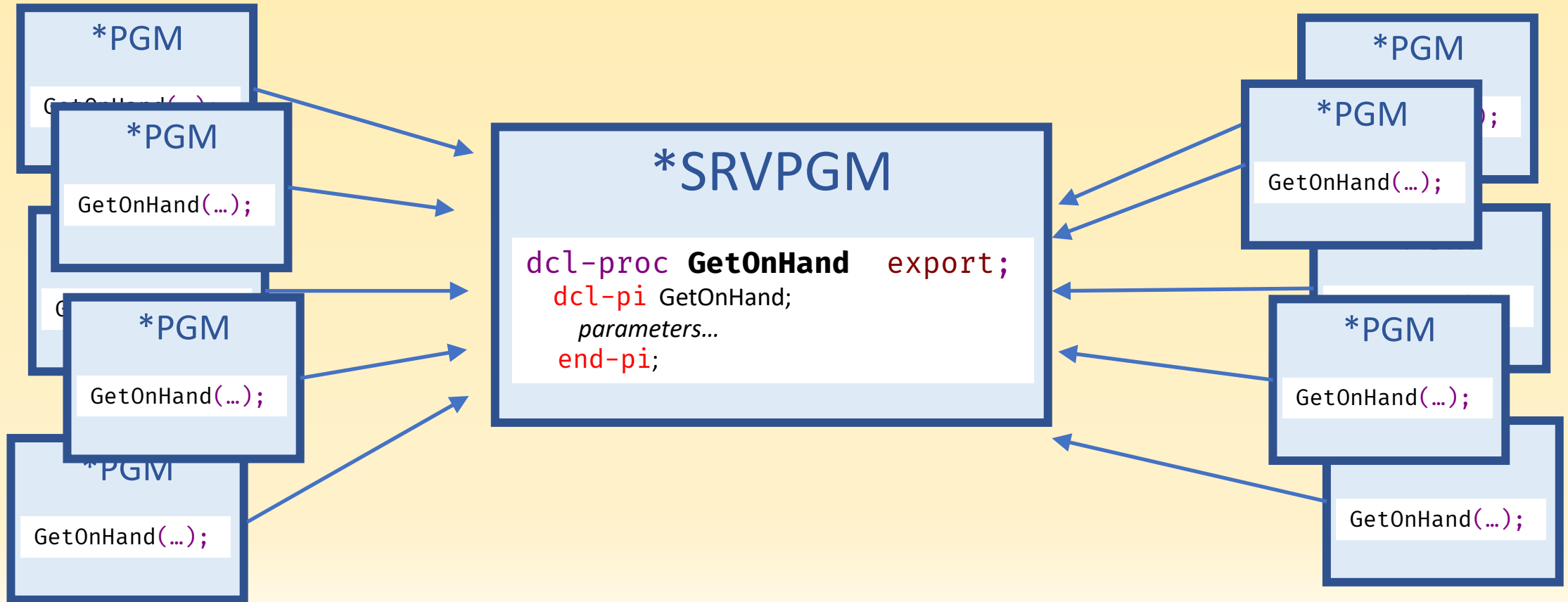


*SRVPGM

```
dcl-proc GetOnHand export;  
  dcl-pi GetOnHand;  
    parameters...  
  end-pi;
```

Alternatives - OPM

- Create a new `*SRVPGM` for on-hand calcs

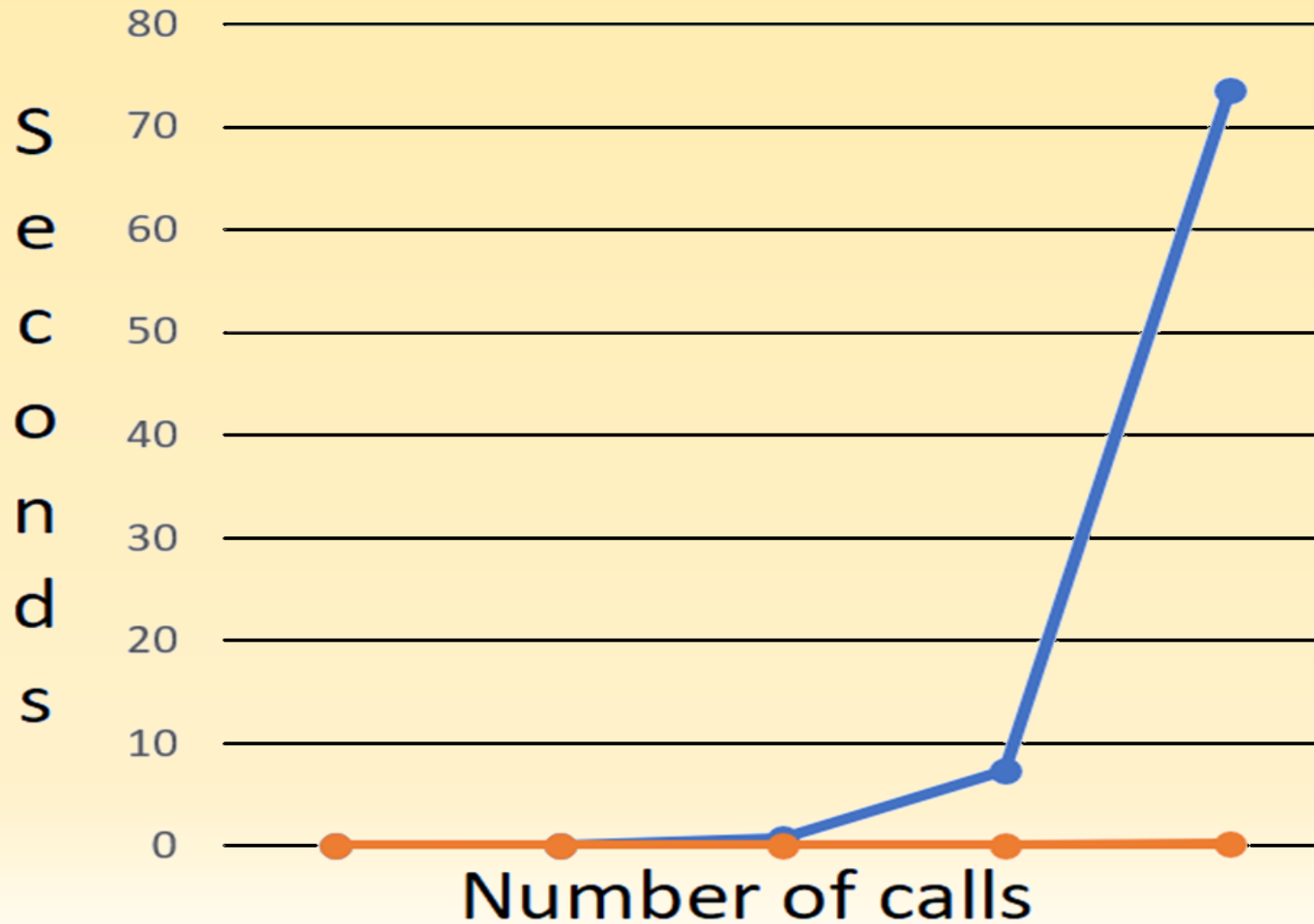


Alternatives - ILE

- Create a service program and BIND by reference
 - It would be very fast
 - It would be modularized
 - Only change the service program next time
- The initial call has some overhead
 - Subsequent calls are much faster than program calls



Alternative - ILE vs OPM



Alternatives – OPM vs ILE

- Dynamic CALL

- Easy to do
- Modular

- 10-character names
- A program for each function
- Overhead of dynamic calls
- Can only return a parameter



- BIND by reference

- Easy(ish) to do
- Modular
- It would be very fast

- Long procedure names
- Many procedures
- Faster calls
- Can return values



Alternatives – the winner

- Service Program
 - Easy
 - Modular
 - Fast
 - One place for reusable code
 - Return values
 - Long names



Creating a procedure

GetOnhand

```
dcl-s weGot packed(7);  
  
CHAIN (loc : item) INVBAL;  
weGot = ONHAND;  
CHAIN (item) REPLACEITM;  
if %found(REPLACEITM);  
    CHAIN (loc : rplitem) INVBAL;  
    weGot += ONHAND;  
endif;
```

GetOnhand

```
dcl-proc GetOnhand;
```

← Procedure name

```
dcl-s weGot packed(7);  
  
CHAIN (loc : item) INVBAL;  
weGot = ONHAND;  
CHAIN (item) REPLACEITM;  
if %found(REPLACEITM);  
    CHAIN (loc : rplitem) INVBAL;  
    weGot += ONHAND;  
endif;
```

```
end-proc;
```

GetOnhand

```
dcl-proc GetOnhand;
  dcl-pi *n packed(7);
    loc char(5) const;
    item char(15) const;
  end-pi;

  dcl-s weGot packed(7);

  CHAIN (loc : item) INVBAL;
  weGot = ONHAND;
  CHAIN (item) REPLACEITM;
  if %found(REPLACEITM);
    CHAIN (loc : rplitem) INVBAL;
    weGot += ONHAND;
  endif;

end-proc;
```

Return type

Input parameters

GetOnhand

```
dcl-proc GetOnhand;
  dcl-pi  *n    packed(7);
    loc    char(5)  const;
    item   char(15) const;
end-pi;


dcl-s weGot packed(7);

CHAIN (loc : item) INVBAL;
weGot = ONHAND;
CHAIN (item) REPLACEITM;
if %found(REPLACEITM);
  CHAIN (loc : rplitem) INVBAL;
  weGot += ONHAND;
endif;

return weGot;

end-proc;
```

Return value



GetOnhand

```
dcl-pr  GetOnhand  packed(7);  
    loc      char(5)  const;  
    item     char(15) const;  
end-pr;
```

```
onhand = GetOnhand( 'LOC' : 'ITEM' );
```

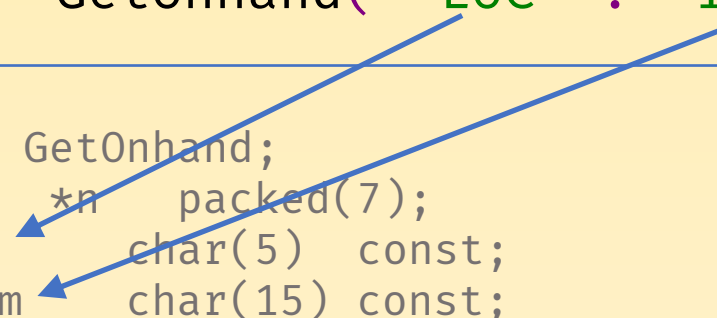
```
dcl-proc  GetOnhand;  
    dcl-pi  *n      packed(7);  
        loc      char(5)  const;  
        item     char(15) const;  
end-pi;  
  
    [...code...]  
  
    return weGot;  
  
end-proc;
```

GetOnhand

```
dcl-pr  GetOnhand  packed(7);  
  loc   char(5)   const;  
  item  char(15)  const;  
end-pr;
```

```
onhand = GetOnhand( 'LOC' : 'ITEM' );
```

```
dcl-proc  GetOnhand;  
  dcl-pi  *n    packed(7);  
  loc    char(5)  const;  
  item   char(15) const;  
end-pi;  
  
  [...code...]  
  
  return weGot;  
  
end-proc;
```

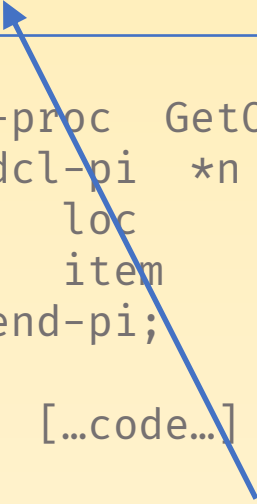


GetOnhand

```
dcl-pr  GetOnhand  packed(7);  
    loc      char(5)  const;  
    item     char(15) const;  
end-pr;
```

```
onhand = GetOnhand( 'LOC' : 'ITEM' );
```

```
dcl-proc  GetOnhand;  
    dcl-pi  *n      packed(7);  
        loc      char(5)  const;  
        item     char(15) const;  
end-pi;  
  
    [...code...]  
  
    return weGot;  
  
end-proc;
```



Creating a service program

Creating a service program

```
dcl-proc  GetOnhand  export ;  
  dcl-pi  *n    packed(7);  
    loc    char(5)  const;  
    item   char(15) const;  
end-pi;  
  
...[code]...  
  
  return weGot;  
  
end-proc;
```

Creating a service program

****free**

ctl-opt nomain;

```
dcl-proc  GetOnhand  export;
  dcl-pi  *n    packed(7);
    loc   char(5)  const;
    item  char(15) const;
end-pi;

...[code]...

return weGot;

end-proc;
```

Creating a service program

****free**

ctl-opt nomain;

/include MYSRVPGM_H ←

```
dcl-proc  GetOnhand  export;
  dcl-pi   *n        packed(7);
    loc    char(5)   const;
    item   char(15)  const;
  end-pi;

  ...[code]...

  return weGot;

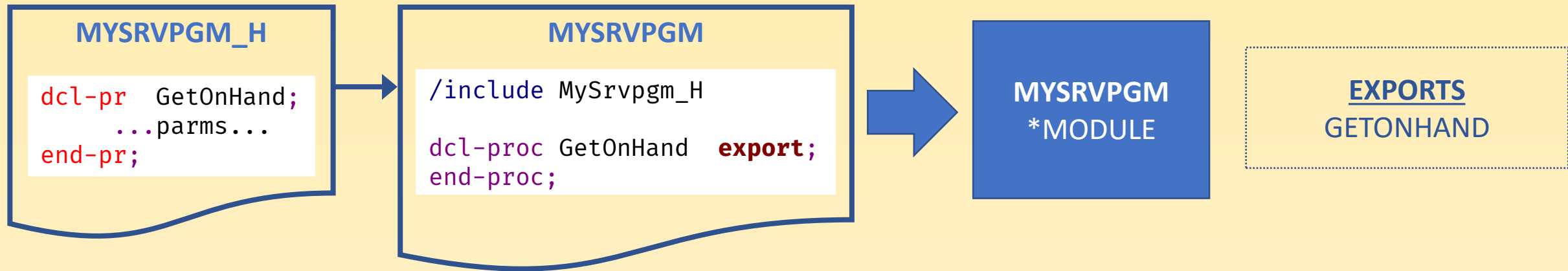
end-proc;
```

****free**

```
dcl-pr  GetOnhand  packed(7);
  loc    char(5)   const;
  item   char(15)  const;
end-pr;
```

Creating a service program

CRTSQLRPGI OBJ(MYSRVPGM) OBJTYPE(*MODULE)



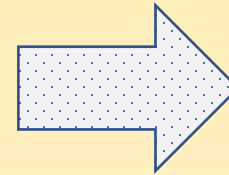
CRTSRVPGM SRVPGM(MYSRVPGM) EXPORT(*ALL)



Creating a service program

Application program

```
**free  
  ctl-opt dftactgrp(*no);  
  
  /include MYSRVPGM_H  
  
onhand = GetOnhand('LOC' : 'ITEM');
```



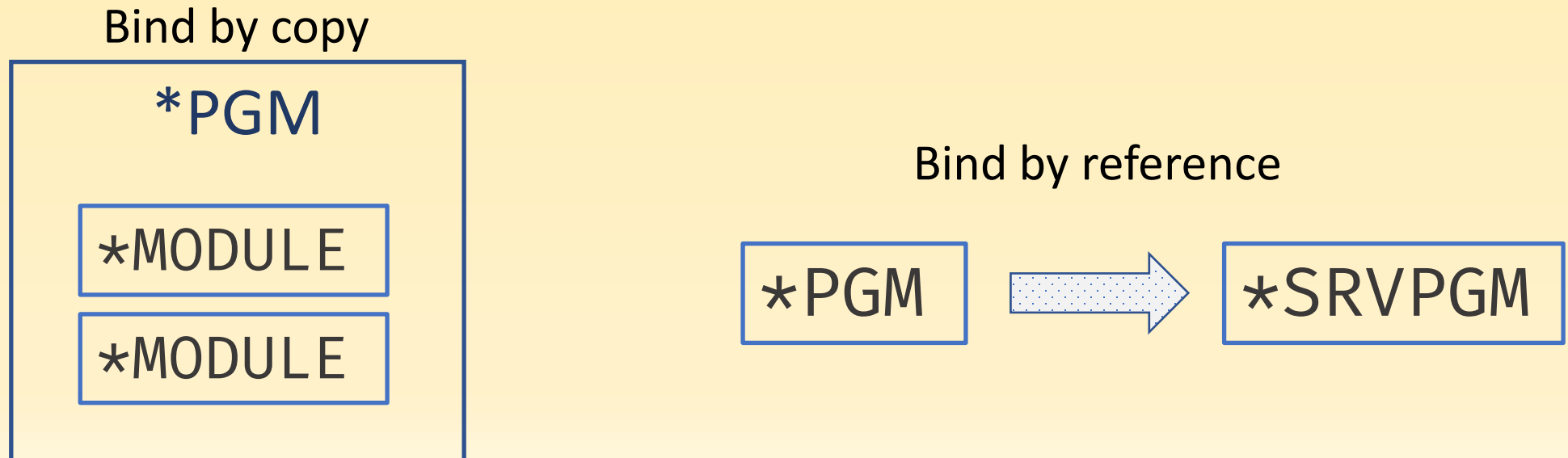
Service program

```
dcl-proc GetOnhand export;  
  dcl-pi  *n    packed(7);  
    loc    char(5)  const;  
    item   char(15) const;  
  end-pi;  
  
  [...code...]  
  
  return weGot;  
  
end-proc;
```

Binding

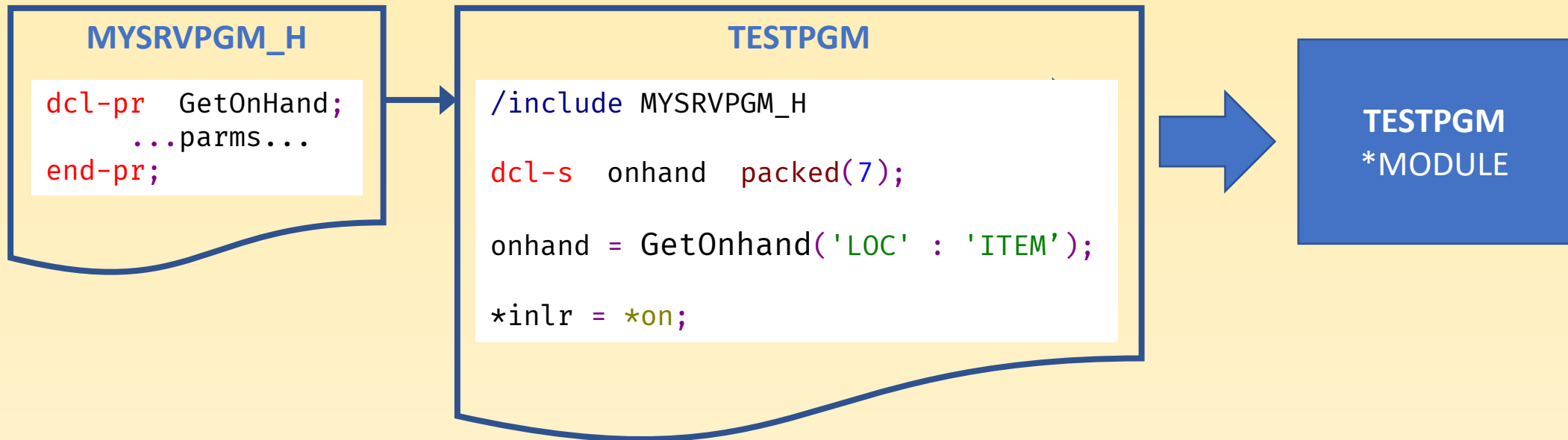
Binding

Binding is the process of creating a runnable ILE program by combining modules and service programs ... resolving symbols passed between them.



Binding

CRTSQLRPGI OBJ(TESTPGM) OBJTYPE(*MODULE)



Binding

```
CRTSQLRPGI OBJ(TESTPGM) OBJTYPE(*MODULE)  
CRTPGM PGM(TESTPGM) BNDSRVPGM(MYSRVPGM)
```



Binding

*SRVPGM

- Write source code
- Create module
- Create service program

*PGM

- Write source code
- Create module
- Create program
BNDSRVPGM to bind all
service programs you need

Binding



Binding – Binding directory

CRTBNDDIR BNDDIR(MYBNDDIR)

	Binding Directory

Binding – Binding directory

CRTBNDDIR BNDDIR(MYBNDDIR)
ADDBNDDIRE BNDDIR(MYBNDDIR) OBJ((MYSRVPGM))

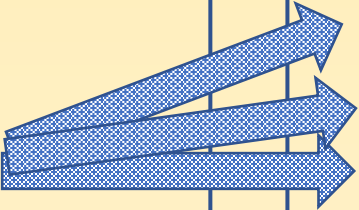
	Binding Directory
	MYSRVPGM

Binding – Binding directory

CRTSQLRPGI OBJ(TESTPGM)

```
**free  
ctl-opt bnddir('MYBNDDIR');  
/include MYSRVPGM_H  
  
dcl-s onhand packed(7);  
  
onhand = GetOnhand('LOC' : 'ITEM');  
  
*inlr = *on;
```

Binding Directory	
	YOURSrvPGM
	OURSrvPGM
	MYSrvPGM
	OTHRsrvPGM



Service Program Review

Creating a service program

MYSRVPGM.SQLRPGLE

```
**free
  ctl-opt nomain;

  /include MYSRVPGM_H ←

dcl-proc GetOnHand export;
  dcl-pi *n packed(7);
    loc char(5) const;
    item char(15) const;
  end-pi;

  ...[code]...

  return weGot;
end-proc;
```

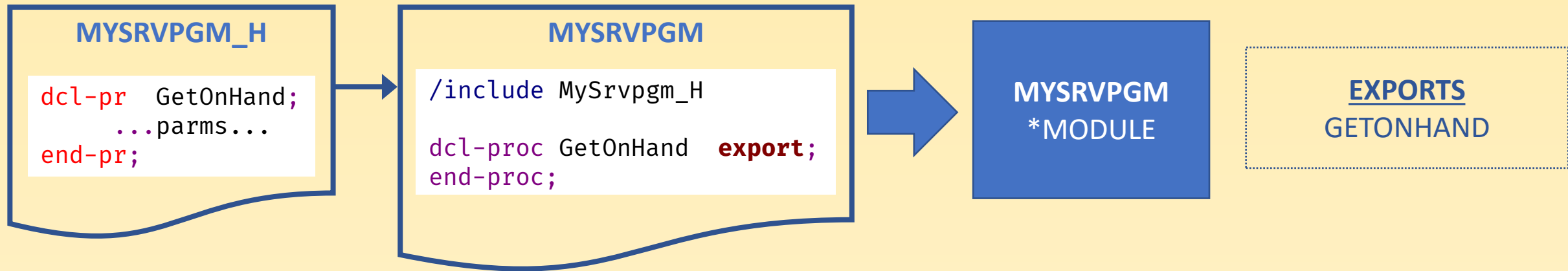
MYSRVPGM_H.RPGLE

```
**free

  dcl-pr GetOnhand packed(7);
    loc char(5) const;
    item char(15) const;
  end-pr;
```

Creating a service program

CRTSQLRPGI OBJ(MYSRVPGM) OBJTYPE(*MODULE)



CRTSRVPGM SRVPGM(MYSRVPGM) EXPORT(*ALL)



Binding – Binding directory

CRTBNDDIR BNDDIR(MYBNDDIR)
ADDBNDDIRE BNDDIR(MYBNDDIR) OBJ((MYSRVPGM))

	Binding Directory
	MYSRVPGM

Using a service program

3 easy steps

1. Use the binding directory
2. Include the prototypes
3. Use the procedures

```
**free  
ctl-opt bnmdir('MYBNDDIR');  
/include MYSRVPGM_H  
  
dcl-s onhand packed(7);  
  
onhand = GetOnhand('LOC':'ITEM');  
  
*inlr = *on;
```

WOOT!



More Procedures

More Procedures

MYSRVPGM.SQLRPGLE

```
**free
  ctl-opt nomain;

  /include MYSRVPGM_H

  dcl-proc GetOnHand      export;

  dcl-proc GetABC        export;
    dcl-pi *n char(3);
    end-pi;
    return 'ABC';
  end-proc;
```

MYSRVPGM_H.RPGLE

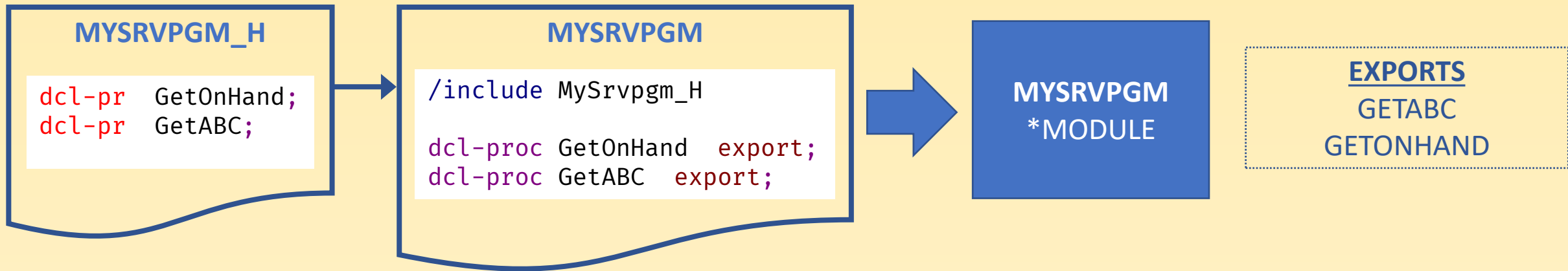
```
**free

  dcl-pr GetOnhand  packed(7);
    loc char(5) const;
    item char(15) const;
  end-pr;

  dcl-pr GetABC char(3);
  end-pr;
```

More Procedures

CRTSQLRPGI OBJ(MYSRVPGM) OBJTYPE(*MODULE)



CRTSRVPGM SRVPGM(MYSRVPGM) EXPORT(*ALL)



More Procedures

3 easy steps

1. Use the binding directory
2. Include the prototypes
3. Use the procedures

TESTPGM2.SQLRPGLE

```
**free  
ctl-opt bnmdir('MYBNDDIR');  
/include MYSRVPGM_H  
  
dcl-s abc char(3);  
  
abc = GetABC();  
  
*inlr = *on;
```

This is easy!



Houston, we have a problem

CALL TESTPGM

Program signature violation.
Error found on CALL command.

Message : Program signature violation.
Cause : The source program TESTPGM
specifies a signature X'0000000000000000C4D5C1C8D5D6E3C5C7'
which is not supported by service program MYSRVPGM.
Recovery : The service program interface
has changed. Re-bind source program TESTPGM.

Dang it! This was supposed to be easy!!



Signatures

Signatures

A signature is generated from

- the names of procedures
- the order in which they are specified

Signatures

MYSRVPGM.SQLRPGLE

```
**free
```

```
ctl-opt nomain;
```

```
/include MYSRVPGM_H
```

```
dcl-proc GetOnhand export;
```

```
...  
end-proc;
```

Signatures

DSPSRVPGM SRVPGM(MYSRVPGM) DETAIL(*SIGNATURE)

Display Service Program Information

Display 1 of 1

```
-  
Service program . . . . . : MYSRVPGM  
  Library . . . . . : PATRICK  
Owner . . . . . : PATRICK  
Service program attribute . . . . . : RPGLE  
Detail . . . . . : *SIGNATURE
```

Signatures:

0000000000000000C4D5C1C8D5D6E3C5C7

Signatures

TESTPGM.SQLRPGLE

```
**free  
  ctl-opt bnddir('MYBNDDIR');  
  /include SRVPGM_H  
  
  dcl-s   onhand   packed(7);  
  
  onhand = GetOnhand(l : i);  
  
  *inlr = *on;
```

Signatures

DSPPGM PGM(TESTPGM) DETAIL(*SRVPGM)

Display Program Information

Display 1 of 1

```
Program . . . . . : TESTPGM      Library . . . . . : PATRICK
Owner . . . . . : PATRICK
Program attribute . . : RPGLE
Detail . . . . . : *SRVPGM
```

Opt	Service Program	Library	Activation	Signature
—	MYSRVPGM	*LIBL	*IMMED	0000000000000000C4D5C1C8D5D6E3C5C7
—	QRNXIE	QSYS	*IMMED	D8D9D5E7C9C540404040404040404040
—	QRNXUTIL	QSYS	*IMMED	D8D9D5E7E4E3C9D34040404040404040
—	QLEAWI	QSYS	*IMMED	44F70FABA08585397BDF0CF195F82EC1

Signatures

Display Service Program Information

Display 1 of 1

Service program : MYSRVPGM

Signatures:

0000000000000000C4D5C1C8D5D6E3C5C7

Display Program Information

Display 1 of 1

Program : TESTPGM

Library : PATRICK

Opt	Service Program	Library	Activation	Signature
_	MYSRVPGM	*LIBL	*IMMED	0000000000000000C4D5C1C8D5D6E3C5C7

0000000000000000C4D5C1C8D5D6E3C5C7

Signatures

MYSRVPGM.SQLRPGLE

```
**free
```

```
ctl-opt  nomain;
```

```
/include MYSRVPGM_H
```

```
dcl-proc GetOnhand      export;
```

```
...
```

```
end-proc;
```

```
dcl-proc GetABC        export;
```

```
...
```

```
end-proc;
```

Signatures

DSPSRVPGM SRVPGM(MYSRVPGM) DETAIL(*SIGNATURE)

Display Service Program Information

Display 1 of 1

```
-  
Service program . . . . . : MYSRVPGM  
  Library . . . . . :   PATRICK  
Owner . . . . . : PATRICK  
Service program attribute . . . . . : RPGLE  
Detail . . . . . : *SIGNATURE
```

Signatures:

00000000000000C4D5C05754431123547

Signatures

Display Service Program Information

Display 1 of 1

Service program : MYSRVPGM

Signatures:

0000000000000000C4D5C05754431123547

Display Program Information

Display 1 of 1

Program : TESTPGM

Library : PATRICK

Opt	Service Program	Library	Activation
—	MYSRVPGM	*LIBL	*IMMED

Signature
0000000000000000C4D5C1C8D5D6E3C5C7

Houston, we have a problem

CALL TESTPGM

Program signature violation.
Error found on CALL command.

Message : Program signature violation.

Cause : The source program TESTPGM
specifies a signature x'0000000000000000C4D5C1C8D5D6E3C5C7'
which is not supported by service program MYSRVPGM.

Recovery : The service program interface
has changed. Re-bind source program TESTPGM.

The truth about binder source

EXPORT(*ALL) is the root of all [binding] evil

Binder source – TLDR;

- Use to control the signature and order of exports
 - Pad signature to 16-characters to remove compile warnings
- Add new procedures to the bottom of the list...ALWAYS
 - Never change the order of the exports
- Use double quotes and match case used in the module
 - Use `DSPMOD DETAIL(*EXPORT)` to see module exports

Current binder source

RTVBNDSRC SRVPGM(MYSRVPGM)

Creates a source member in QSRVSRC with the same name as the service program

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE(X'00000000000000C4D5C05754431123547' )
/*****
/*      *SRVPGM      MYSRVPGM      PATRICK      05/03/20  16:51:38      */
/*****
  EXPORT SYMBOL("GETONHAND")
  EXPORT SYMBOL("GETABC")
ENDPGMEXP
```

New signature

Change the signature to "MYSRVPGM V1.0"

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE("MYSRVPGM V1.0")  
  EXPORT SYMBOL("GETONHAND")  
  EXPORT SYMBOL("GETABC")  
ENDPGMEXP
```

```
UPDSRVPGM SRVPGM(MYSRVPGM) MODULE(MYSRVPGM)  
  EXPORT(*SRCFILE)
```

Current export signature : MYSRVPGM V1.0

x'D4E8E2D9E5D7C7D440E5F14BF0404040'

New signature

Display All Messages

Job . . : QPADEV0001 User . . : PATRICK Number . . . : 132411

4 > UPDSRVPGM SRVPGM(MYSRVPGM) MODULE(MYSRVPGM) EXPORT(*SRCFILE)

Binder source line 1: ***** Signature padded to 'MYSRVPGM V1.0'

1 warnings were issued from binder language compilation.

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE("MYSRVPGM V1.0  ")
EXPORT SYMBOL("GETONHAND")
EXPORT SYMBOL("GETABC")
ENDPGMEXP
```

New Signature

MYSRVPGM

```
dcl-proc GetOnhand      export;  
    ...  
end-proc;  
  
dcl-proc GetABC         export;  
    ...  
end-proc;  
  
dcl-proc GetXYZ         export;  
    ...  
end-proc;
```

Current export signature : MYSRVPGM V1.0

Current export signature : MYSRVPGM V1.0

Current export signature : MYSRVPGM V1.0

Order of exports

- Binding is based on the order of exports (not name!)
- The module has exports
 - In alphabetical order
- The service program has exports
 - In the order listed in the binder source
- The order of procedures in the source has no bearing

Order of exports

MYSRVPGM.SQLRPGLE

```
dcl-proc GetOnhand    export;  
dcl-proc GetABC      export;  
dcl-proc GetXYZ      export;
```

MYSRVPGM.BND

```
EXPORT SYMBOL("GETONHAND")  
EXPORT SYMBOL("GETABC")  
EXPORT SYMBOL("GETXYZ")
```

*MODULE

EXPORTS
GETABC
GETONHAND
GETXYZ

*SRVPGM

EXPORTS
GETONHAND
GETABC
GETXYZ

Order of exports

TESTPGM3.SQLRPGLE

```
**free
```

```
ctl-opt bnmdir('MYBNDDIR');
```

```
/include MYSRVPGM_H
```

```
dcl-s onhand packed(7);
```

```
dcl-s rtnval char(3);
```

```
onhand = GetOnhand('LOC' : 'ITEM'); ← 1234567
```

```
rtnval = GetXYZ(); ← "XYZ"
```


```
rtnval = GetABC(); ← "ABC"
```

```
*inlr = *on;
```


Order of exports

MYSRVPGM.SQLRPGLE

```
dcl-proc GetXYZ      export;  
dcl-proc GetOnhand  export;  
dcl-proc GetABC     export;
```



MYSRVPGM.BND

```
EXPORT SYMBOL("GETONHAND")  
EXPORT SYMBOL("GETABC")  
EXPORT SYMBOL("GETXYZ")
```

*MODULE

EXPORTS
GETABC
GETONHAND
GETXYZ

*SRVPGM

EXPORTS
GETONHAND
GETABC
GETXYZ

Order of exports

TESTPGM3.SQLRPGLE

```
**free
  ctl-opt bnmdir('MYBNDDIR');
  /include MYSRVPGM_H
  dcl-s onhand packed(7);
  dcl-s rtnval char(3);

  onhand = GetOnhand('LOC' : 'ITEM'); ← 1234567
  rtnval = GetXYZ(); ← "XYZ"
  rtnval = GetABC(); ← "ABC"

  *inlr = *on;
```


Order of exports

MYSRVPGM.SQLRPGLE

```
dcl-proc GetXYZ      export;  
dcl-proc GetOnhand  export;  
dcl-proc GetABC     export;
```

MYSRVPGM.BND

```
EXPORT SYMBOL("GETONHAND")  
EXPORT SYMBOL("GETXYZ")  
EXPORT SYMBOL("GETABC")
```



*MODULE

EXPORTS
GETONHAND
PROCABC
PROXYZ

*SRVPGM

EXPORTS
GETONHAND
GETXYZ
GETABC



Order of exports

TESTPGM3.SQLRPGLE

```
**free
```

```
ctl-opt bnmdir('MYBNDDIR');
```

```
/include MYSRVPGM_H
```

```
dcl-s onhand packed(7);
```

```
dcl-s rtnval char(3);
```

```
onhand = GetOnhand('LOC' : 'ITEM'); ← 1234567
```

```
rtnval = GetXYZ(); ← "ABC"
```

```
rtnval = GetABC(); ← "XYZ"
```

```
*inlr = *on;
```

ALLUPPERCASEISHARDTOREAD

ALLUPPER vs MixedCase

THISISAREALLYLONGNAMEFORAPROCEDURE

ThisLongNameIsAnEvenLongerNameForAProcedure

ALLUPPER vs MixedCase

MYSRVPGM_H.RPGLE

```
dcl-pr GETONHAND  packed(7)  extproc( 'GetOnHand' );  
    inLoc  char(5)  const;  
    inItem char(15) const;  
end-pr;
```

```
dcl-pr GetABC  char(3)  extproc(*dclcase);  
end-pr;
```

```
dcl-pr GetXYZ  char(3)  extproc(*dclcase);  
end-pr;
```

*MODULE

EXPORTS
GetOnHand
GetABC
GetXYZ

ALLUPPER vs MixedCase

MYSRVPGM.BND

```
EXPORT SYMBOL("GETONHAND")  
EXPORT SYMBOL("GETABC")  
EXPORT SYMBOL("GETXYZ")
```

Service program MYSRVPGM not updated.

Binder source line 2: ***ERROR Symbol not defined: 'GETONHAND'

Binder source line 3: ***ERROR Symbol not defined: 'GETABC'

Binder source line 4: ***ERROR Symbol not defined: ' GETXYZ'

ALLUPPER vs MixedCase

MYSRVPGM.BND

```
EXPORT SYMBOL("GetOnHand")  
EXPORT SYMBOL("GetABC")  
EXPORT SYMBOL("GetXYZ")
```

*SRVPGM

EXPORTS
GetOnHand
GetABC
GetXYZ

ALLUPPER vs MixedCase

TESTPGM3.SQLRPGLE

```
**free
```

```
ctl-opt bnmdir('MYBNDDIR');
```

```
/include MYSRVPGM_H
```

```
dcl-s rtnval char(3);
```

```
rtnval = GETXYZ(); ← "XYZ"
```

```
rtnval = getxyz(); ← "XYZ"
```

```
rtnval = gEtXyZ(); ← "XYZ"
```

```
rtnval = GeTxYz(); ← "XYZ"
```

```
*inlr = *on;
```

Review



Alternatives

- OPM

- Hard-code the programs 🙄😱🤢🐱😞
- Dynamic program calls 😐

- ILE

- Modules + bind by copy 🙄😱🐱😞
- Service program + bind by reference 😁🏆

Writing a service program

```
**free
```

```
ctl-opt nomain;  
/include MYSRVPGM_H
```

```
dcl-proc GetOnhand export;  
  dcl-pi *n packed(7);  
    loc char(5) const;  
    item char(15) const;  
  end-pi;
```

```
...[get onhand code]...
```

```
  return weGot;
```

```
end-proc;
```

```
**free
```

```
dcl-pr GetOnhand packed(7) extproc(*dclcase);  
  loc char(5) const;  
  item char(15) const;  
end-pr;
```

Creating the Binder Source

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE("MYSRVPGM V1.0")  
  EXPORT SYMBOL("GetOnHand")  
  EXPORT SYMBOL ("GetABC")  
ENDPGMEXP
```

Creating a service program

CRTSQLRPGI OBJ(MYSRVPGM) OBJTYPE(*MODULE)

```
**free  
ctl-opt nomain;  
/include MYSRVPGM_H  
  
dcl-proc GetOnhand export;  
dcl-proc GetABC export;
```

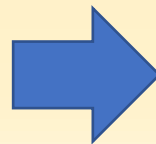


MYSRVPGM
*MODULE

EXPORTS
GetABC
GetOnHand

CRTSRVPGM SRVPGM(MYSRVPGM) EXPORT(*SRCFILE)

MYSRVPGM
*MODULE



MYSRVPGM
*SRVPGM

EXPORTS
GetOnHand
GetABC

Create the Binding directory

```
CRTBNDDIR      BNDDIR(MYBNDDIR)
ADDBNDDIRE    BNDDIR(MYBNDDIR)  OBJ((MYSRVPGM))
```

	Binding Directory
	MYSRVPGM

Using a Service Program

3 easy steps

1. Use the binding directory
2. Include the prototypes
3. Use the procedures

TESTPGM.SQLRPGLE

```
**free  
ctl-opt bnmdir('MYBNDDIR');  
/include MYSRVPGM_H  
  
dcl-s abc char(3);  
  
abc = GetABC();  
  
*inlr = *on;
```