**ILE: Integrated Language environment in AS400**

1. This is set of tools or system support to enhance program development in AS400
2. **Benefits: -**

  Binding, modularity, reusable components, common runtime services, coexistence, and a source debugger. They also include better control over resources, better control over language interactions, better code optimization, a better environment for C, and a foundation for the future.

1. **Modules and procedures:**

A module is an non runnable object of type \*MODULE When we compile a module's source code by taking **option 15(against 14 for programs**), we don't get an executable file, but a unit that when connected to other units results in the executable object. A module can have on or more procedures. A procedure is like a function

1. **Exports**

An export is the name of a procedure or data item, coded in a module object, that is available for use by other ILE objects. The export is identified by its name and its associated type, either procedure or data.

An export can also be called a definition.

1. **Imports**

An import is the use of or reference to the name of a procedure or data item not defined in the current module object. The import is identified by its name and its associated type, either procedure or data.

An import can also be called a reference

1. **Program entry procedure (PEP)**

A program entry procedure is the compiler-generated code that is the entry point for an ILE program on a dynamic program call. It is similar to the code provided for the entry point in an OPM program.

A program entry procedure/module **will not have \*NOMAIN** in H-SPEC, while other modules with ILEPGM **will have \*NOMAIN in H-SPEC**.

1. **EXTPGM :** The EXTPGM keyword indicates that the prototype represents a dynamic call to a program.
2. **EXTPROC** The EXTPROC keyword indicates the external name of the procedure whose prototype is being defined. ... It should point to a procedure whose return value and parameters are consistent with the prototype definition. When a prototype is specified for a procedure, the EXTPROC keyword is specified for the prototype.
3. CALLP and CALLB

The CALLP operation is used to call prototyped procedures or programs. This is generally used to call procedures.

**Free format program call:-** In free format programs we cannot directly call programs , we need to define prototype and use EXTPGM keyword to make that prototype refer external program and then you can invoke program call using prototype call.

The CALLB operation is used to call bound procedures written in any of the ILE language. This is generally used to call Modules

**Difference between CALL, CALLB and CALLP**  
CALL is a dynamic call where the control will be transferred when the program is executed. (Control will be transfer the another program (run time) so it is dynamic call).  
Where as CALLB and CALLP are static calls. A module is a non-executable program and it contains one or more procedures. If you have modules without procedure then it means that it is having only one default procedure and in case we can use CALLB.  
A module is having more than one procedure then we can give explicitly the procedure name to be called in case of CALLP out of these three CALLP is the most efficient one. (Using the CALLB, CALLP a program or module is bind in the program so it is static.)

**Example:**

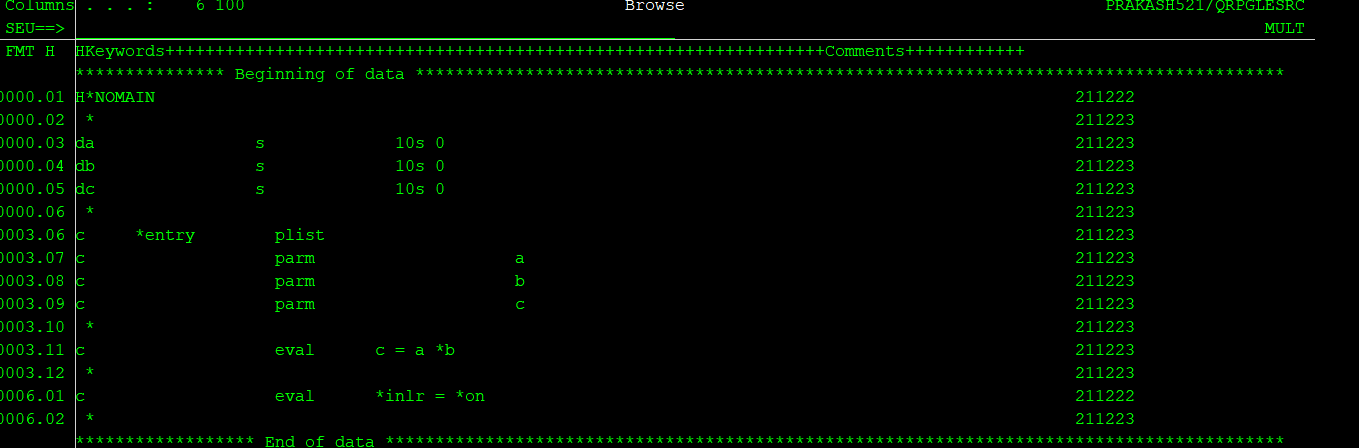
Consider a below ILE Program with 2 Modules

Module 1 : PROCMOD which has 2 procedures in IT

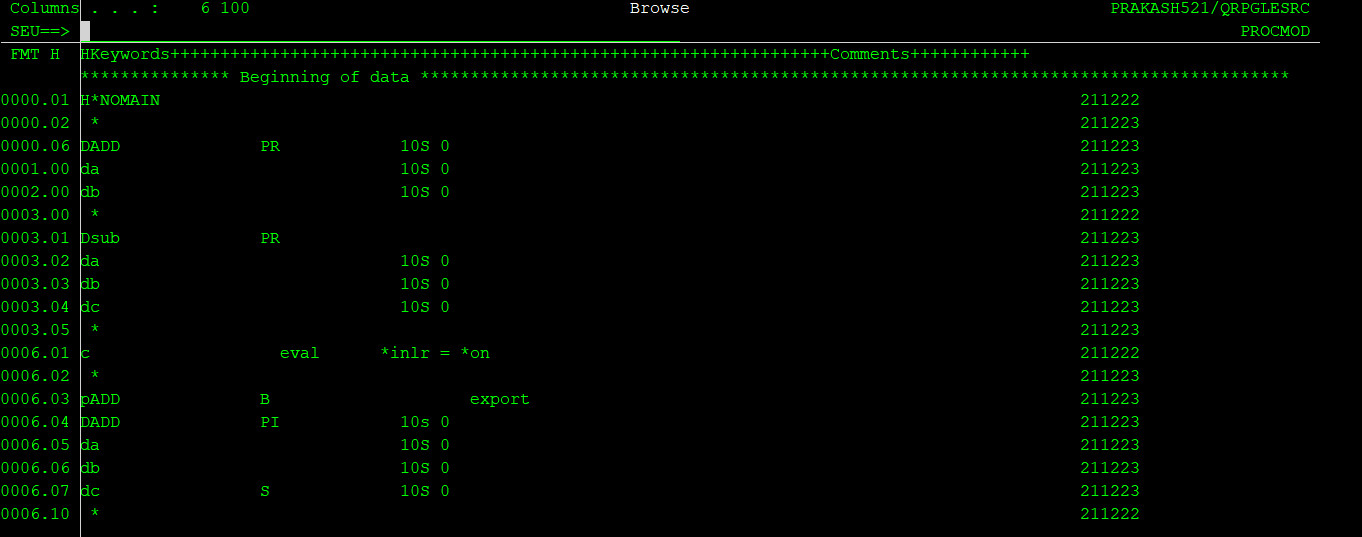
Module 2 : Has no procedures , just calculation

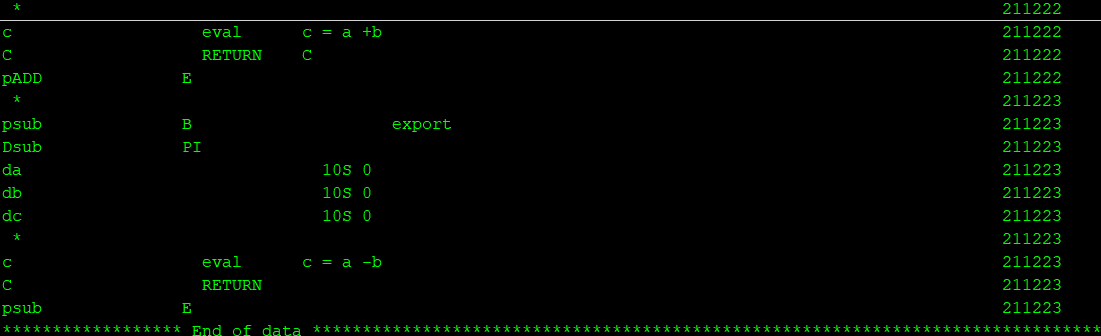
)

**MULT Module:-**

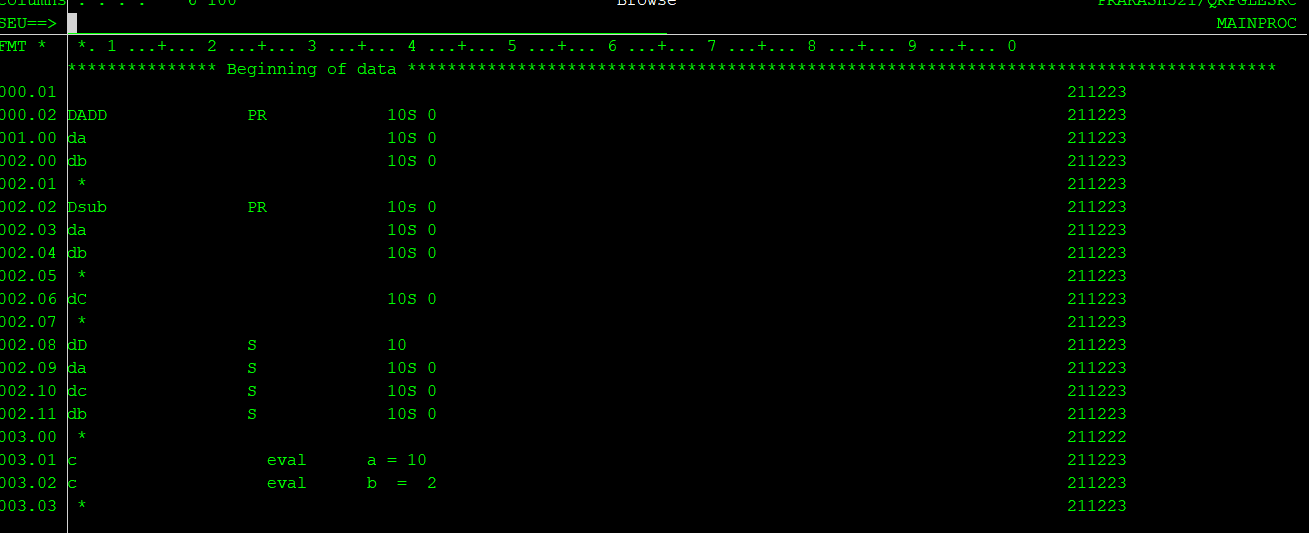


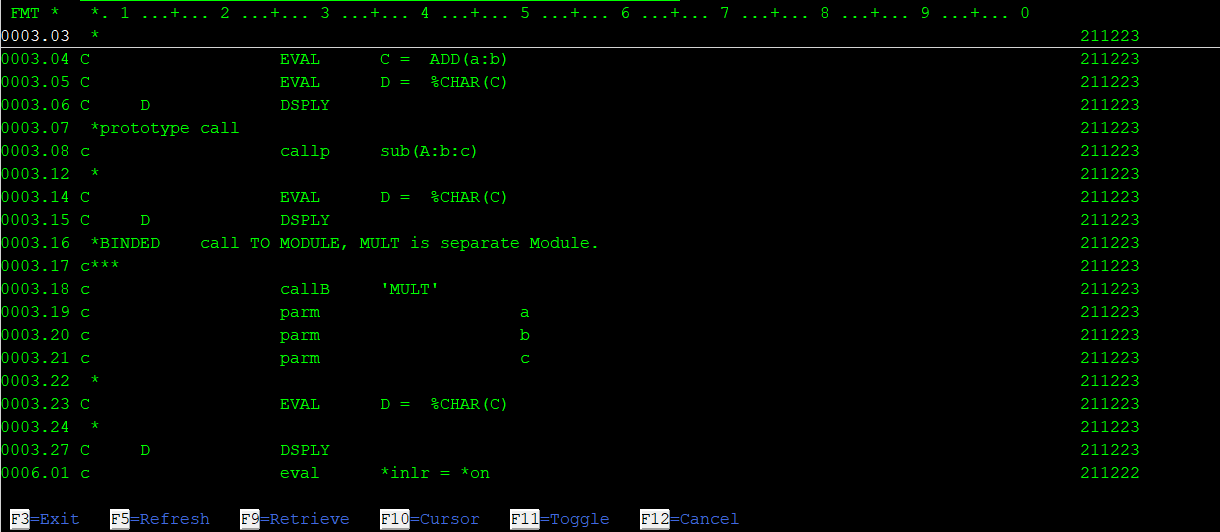
**PROCMOD**





**MAINPROC Module :**





**Create command to create ILEPROGRAM from these modules.**

CRTPGM PGM(CALILEPGM) MODULE (MAINPROC PROCMOD MULT)

**Static binding & Dynamic binding**

 .

·         Dynamic binding means that neither program knows about the other until the call takes place. In that sense it is dynamic, because all the binding is happening right at the time the program is called. Whenever you call one program from another using the CALL operation, you are dynamically binding those programs.

·       Static binding, on the other hand, is the method of binding two objects together well before they ever use each other. This type of binding is static, in that the objects are bound together and the relationship between those bound objects remains consistent.

**Types of static Binding**:

1. **Bind by copy:** When we create ILEPGM by directly binding modules, modules are copied into ILEPGM and ILEPGM is created. Suddenly after creating ILEPGM if module gets deleted still it won’t have any impact as Modules are already copied.
2. **Bind by reference:** In this case address/reference is stored while creating ILEPGM. ILE programs which are created by binding service programs are example of bind by reference

**Service programs in AS400**

A service program is a collection **of runnable procedures** and available data items that are used by other ILE program objects and service programs.

CRTSRVPGM is used to create service programs.

Service program is not a actual program, Like Modules it can’t be called by itself. It has to be binded to a ILE program,

This is an AS400 object of type \*SRVPGM

**Binding directory**

A **binding directory** lists the names of modules and service programs that you may need when creating an ILE program or service program

It can be created by CRTBNDDIR command.

DSPBNDDIR can be used to view entries in that binding directory

ADDBNDDIR command can be used to to add entries to Binding directory.

WRKBNDDIR command can be used to work with binding directory. This is more convenient command to view/add/delete entries from Binding directory.

It is an AS400 object of type \*BNDDIR

Binding directories are optional. The reasons for using binding directories are convenience and program size.

They offer a convenient method of listing the modules or service programs that you may need when creating your own ILE program or service program. For example, one binding directory may list all the modules and service programs that provide math functions. If you want to use some of those functions, you specify only the one binding directory, not each module or service program you use.

# **Binder Language**

The **binder language** is a small set of nonrunnable commands that defines the exports for a service program.

Binder language consists of a list of the following commands:

1. Start Program Export (STRPGMEXP) command, which identifies the beginning of a list of exports from a service program
2. Export Symbol (EXPORT) commands, each of which identifies a symbol name available to be exported from a service program
3. End Program Export (ENDPGMEXP) command, which identifies the end of a list of exports from a service program

Figure 1. Example of Binder Language in a Source File

1. STRPGMEXP PGMLVL(\*CURRENT) LVLCHK(\*YES)
2. .
3. .
4. EXPORT SYMBOL(p1)
5. EXPORT SYMBOL('p2')
6. EXPORT SYMBOL('P3')
7. .
8. .
9. ENDPGMEXP
10. .
11. .
12. .
14. STRPGMEXP PGMLVL(\*PRV)
15. .
16. .
17. EXPORT SYMBOL(p1)
18. EXPORT SYMBOL('p2')
19. .
20. .
21. ENDPGMEXP

.

**Source type of binder source is BNDSRC**

Using the Retrieve Binder Source (RTVBNDSRC) command, you can generate the binder language source based on exports from one or more modules or service programs.

In below case all exports defined in Modules are taken, everything is exported

**CRTSRVPGM SRVPGM(CALCSRVPGM) MODULE(PROCMOD) EXPORT(\*ALL)**

In below case exports are taken from Binder source in QSRVSRC ,Source member CALCSRVPGM

**CRTPGM PGM(CALCILEPGM) MODULE(MAINPROC MULT) BNDSRVPGM((CALCSRVPGM))**

**In case if any of the existing procedure logic is changed signature wont be changed, hence no recompilation is needed for ILE programs using this service program.**

**In case when new procedure is added and exported , signature will be changed we can maintain different signatures in binder source and then with that programs not needing this new procedure need not be compiled they will refer to old signature and programs needing this new procedure only needs to be recompiled , they will refer new signature and new procedure will be available to them.**