**Fully free format**

Special directive \*\*FREE indicates that the entire source member contains fully free-form code. Fully free-form code can appear in any column, from column 1 to the end of the line. There is no practical limit on the length of a source line in fully free-form source.

\*\*FREE may only be specified in column 1 of the first line in the source. The remainder of the line must be blank.

**File declaration :**

DCL-F FILENAME DISK **(Default to \*INPUT)**

DCL-F FILENAME DISK KEYED USAGE(\*UPDATE) Here **we can do INPUT /update but not delete**

**(You can delete in above example, however in specification declaration just U will be enough for delete)**

DCL-F FILENAME DISK KEYED USAGE(\*DELETE) Here **we can do INPUT /update and delete**

dcl-f employee disk usage(\*UPDATE:\*output) keyed ; **(Here update and write operation can be performed , but not delete)**

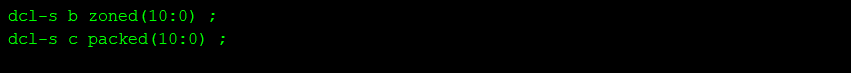
dcl-f testpf usage(\*update) keyed rename(testpfr:newname)

**using rename keyword in F-SPEC**

dc-f printerf printer; **Example for printer file**

dcl-f dsppf workstn sfile(sfl01: rrn) **; Example for subfile**

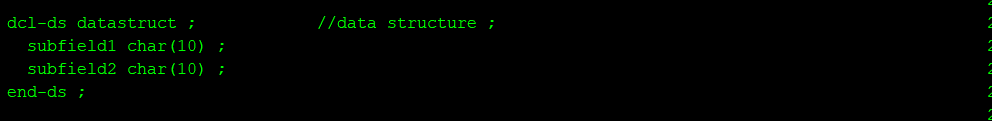
**Standalone Variable declaration in Free Format**



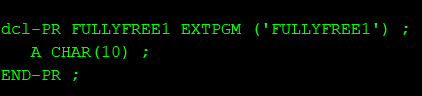
Array declaration in full free format.



**Data structure declaration in Free Format**



**Declaring procedure prototype in Free format:**



Call fullyfree1

Parm a

Dcl-pr free\_Call extpgm(‘pgmnam’)

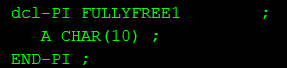
A char(10)

End-pr

FREE\_Call (fld1);

EXTPGM denotes that this procedure call will invoke external program.

**Declaring procedure interface in Free format:**



**How to add parameters to programs in free format?**

Declare a prototype and interface with EXTPGM in prototype referring to same program name.

**Calling a program in free format?**

Declare a prototype with extpgm keyword pointing to program which needs to be called and then use prototype call.

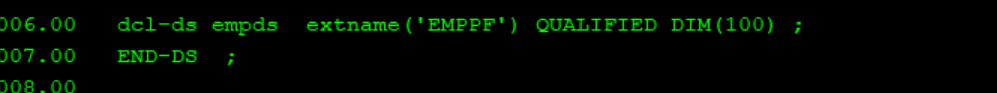
Prototype declaration DCL-PR Start of prototype and END-PR end of prototype

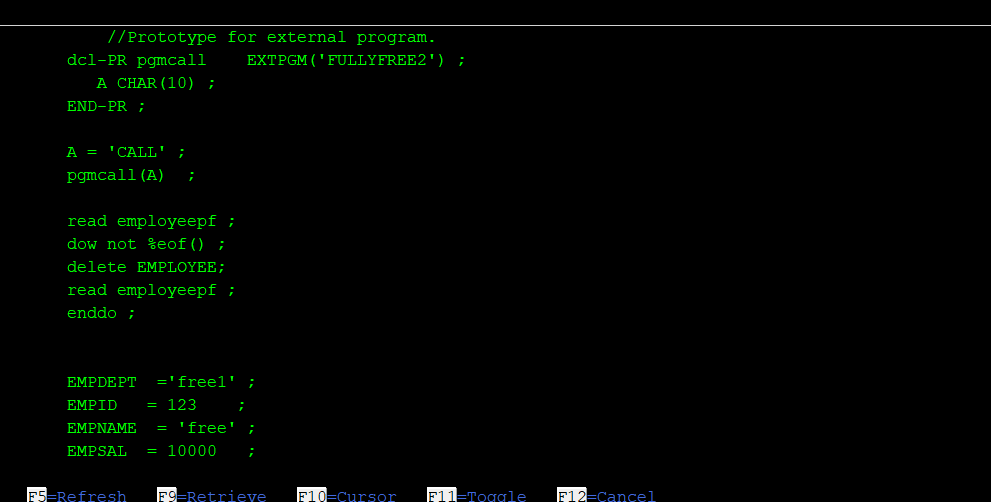
**Example : -**

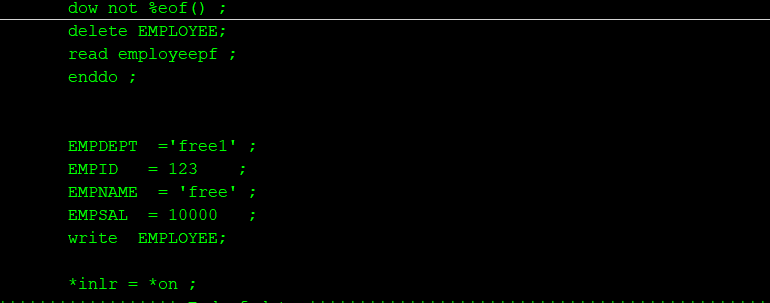
**1ST programm FULLYFREE1 calls FULLYFREE2**



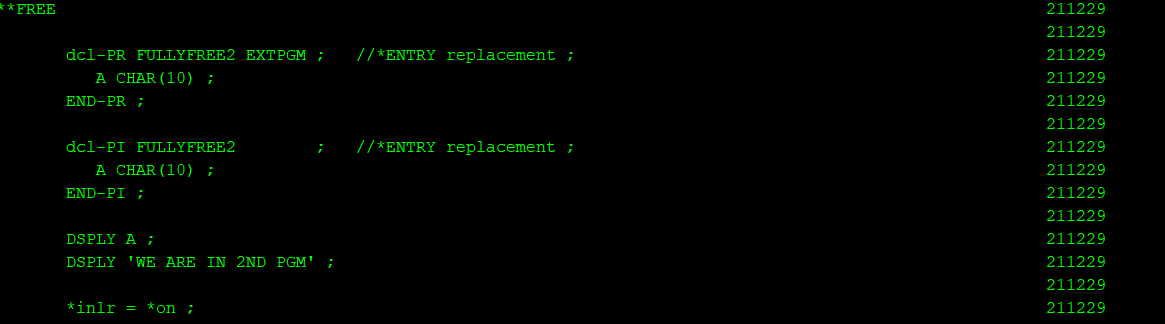
**EXTERNAL data structure defintion which is qualified and group of elements.**

****





**FULLYFREE2**



**Control specification definition(H spec equivalent)**

It is defined with CTL-OPT

**CTL-OPT DFTACTGRP(\*NO) ACTGRP(‘PAYMENTS’)**

**OPTION(\*SRCSTMT:\*NODEBUGIO**

**Chain/reade in free format**

Chain (keyfield1:keyfield2:keyfield2) filename

Reade (keyfield1:keyfield2:keyfield2) filename

***%KDS in RPGLE***

%KDS(data-structure-name{:num-keys})

*This is basically equivalent of KLIST in free format.*

*First data structure is defined with LIKEREC keyword and \*key .*

*Then it is used to search record in file. Number of keys parameter denotes total number of keys in file , this is optional.*

gure 1. Example of Search on Keyed Input/Output Operations

A..........T.Name++++++RLen++TDpB......Functions++++++++++++++++++

A R CUSTR

A NAME 100A

A ZIP 10A

A ADDR 100A

A K NAME

A K ZIP

FFilename++IPEASF.....L.....A.Device+.Keywords+++++++++++++++++++++++++

Fcustfile if e k disk rename(CUSTR:custRec)

DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++

D custRecKeys ds likerec(custRec : \*key)

D numKeys s 10i 0

...

/free

// custRecKeys is a qualified data structure

custRecKeys.name = customer;

custRecKeys.zip = zipcode;

// The \*KEY data structure is used as the search argument for CHAIN

chain %kds(custRecKeys) custRec;

// The number of keys can be a constant

chain %kds(custRecKeys : 2) custRec;

// The number of keys can be a variable or an expression

numKeys = 1;

chain %kds(custRecKeys : numKeys) custRec;

chain %kds(custRecKeys : numKeys + 1) custRec;

/end-free

***Sub Procedure declaration logic in free format.***

P spec B and P spec E is replaced by DCL-PROC and END-PROC

***Example:***

DCL-PROC getCurrentUser;

DCL-PI \*N CHAR(10) END-PI;  5

DCL-S currentUser CHAR(10) INZ(\*USER);

RETURN currentUser;

END-PROC;