Guía de desarrollo, Anexo 13

Normas de codificación PL1

Autor: Oficina de Calidad

GERENCIA INFORMÁTICA

JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

Índice de Ilustraciones y Tablas

1 Introducción

1.1 Objetivo

El objetivo de este documento es establecer las normas para el desarrollo de sistemas de información que requieran de programación PL1 en la Gerencia de Informática de la Dirección General de Tráfico.

1.2 Audiencia

Este documento está dirigido a todas las personas que colaboren en labores relacionadas con la gestión, desarrollo, auditoría, implantación y explotación de los sistemas de información de la gerencia de informática de la Dirección General de Tráfico; y en especial a aquellas personas que requieran la realización del desarrollo utilizando el lenguaje de programación PL1.

1.3 Glosario

Los términos y acrónimos que se utilizan en este documento y en el resto de documentos de la guía se encuentran recogidos por orden alfabético en el Anexo 30. Glosario con el objetivo de facilitar su lectura y comprensión.

2 Criterios de Codificación

2.1 Criterios básicos

De base para todos los criterios de codificación, el programa deberá estructurarse y codificarse siguiendo un orden lógico (Programación estructurada Top-Down)

```
IXXXXXX: PROC(PARAMETROS) OPTIONS(PARAMETROS);
```

```
      DCL VARIABLES...
      DCL CONSTANTES...
      DCL XXXXXXX...

/*-------------------------------------------------------------------*/
/* LOGICA DEL PROGRAMA                                               */
/*-------------------------------------------------------------------*/
      CALL INICIO_PROGRAMA;
      CALL PROCESO_PRINCIPAL;
      CALL FINAL_PROGRAMA;
/*-------------------------------------------------------------------*/

      INICIO_PROGRAMA: PROC;
            ...
      END INICIO_PROGRAMA;


      PROCESO_PRINCIPAL: PROC;
            ...
      END PROCESO_PRINCIPAL;


      FINAL_PROGRAMA:
            ...
            RETURN;
      END FINAL_PROGRAMA;

   END IXXXXXX;
```

La estructura del programa, deberá contener los tres procedimientos básicos:

INICIO_PROGRAMA

Uso: Inicialización de variables, validación de datos de entrada y apertura de ficheros (en caso de existir).

PROCESO_PRINCIPAL

Uso: Lógica del programa. Los procedimientos deben tener funciones monofuncionales, claras y especificas. Un proceso de cálculo no debería contener sentencias de lectura, o un procedimiento de impresión de un listado no debe contener actualizaciones a bases de datos, etc. así como también deben evitarse aquellos procedimientos largos.

Se deben utilizar niveles de indentación claros. Utilizando para ello el mismo número de espacios para todo el programa aprovechando los DO y END de las sentencias.

```
DO WHILE (¬EOF)
  |  IF ..................
  |   THEN DO;
  |     | SELECT();
  |     | | WHEN(1)
  |     | |  DO;
  |     | |  | SENTENCIA
  |     | |  END;
```

```
|     | | OTHER
|     | |  DO
|     | |  END
|     | END;
|     END;
|    ELSE DO;
|     | SENTENCIA
|    END;
END;
```

FINAL_PROGRAMA

Uso: Creación de mensajes de salida, estadísticas, cierre de ficheros (en caso de existir) y la única terminación del programa.

El nombre de estos tres procedimientos básicos, en el caso de contener cualquiera de ellos una sola funcionalidad, puede ser otro que identifique la funcionalidad a implementar.

2.2 Uso de includes

Los includes sólo deben contener definiciones de estructuras de datos.

Se crearán includes para todas aquellas estructuras de datos que se vayan a utilizar en más de un módulo de la aplicación.

Debe haber un include con la definición relativa a los parámetros, necesaria por cada módulo llamado.

Ejemplo de include con la definición de los parámetros para llamar a un módulo:

```
/*====================================================================*/
 /* INCLUDE    : KMHFCFM                                               */
 /* APLICACION: COMPONENTES COMUNES                                    */
 /* MODULO     : CMHFCFM                                               */
 /*====================================================================*/
 /* DEFINICION DCL ENTRY                                               */
 /*--------------------------------------------------------------------*/
    DCL CMHFCFM    ENTRY(CHAR(08),  /* FECHA DE ENTRADA               */
                         CHAR(03),  /* FORMATO FECHA DE ENTRADA       */
                         CHAR(08),  /* FECHA DE SALIDA                */
                         CHAR(01)); /* CODIGO RESULTADO               */
 /*====================================================================*/
 /* DEFINICION DCL PARAMETROS                                          */
 /*------------------------------------------------------------------- */
```

```
    DCL  FECHA            CHAR(08),
         FORMATO          CHAR(03),
         FECHA_RESULT     CHAR(08),
         RESULT           CHAR(01);
/*================================================================*/
```

2.3 Nomenclatura de los componentes

2.3.1 Programas

Los programas deberán nombrarse de acuerdo a la siguiente nomenclatura de 7 caracteres: Wxyyzzz

Nomenclatura	Descripción	Opciones	
W	Identificador del tipo de programa	I: Online B: Batch C: Común (programas comunes generales en la instalación proporcionados por la Calidad).	
X	Identificador de la función del programa	Si I: Online	N: Negocio D: Datos T: Teleproceso
		Si B: Batch	M: Módulo P: Programa
		Si C: Común	M: Módulo
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas SA: Sanciones VE: Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos	

ZZZ	Identificador del programa de 3 posiciones, a elección por la aplicación.	

Tabla 1. Nomenclatura de programas.

Los programas con función de Negocio (**N**) no deberán tener ningún tipo de acceso a Base de Datos, estos deberán realizarse por medio de los programas con función de Datos (**D**).

2.3.2 Includes

Los includes deberán nombrarse de acuerdo a la siguiente nomenclatura de 8 caracteres:

WXyyzzzz

La nomenclatura de los includes dependerá su naturaleza. La nomenclatura para includes de definición de estructura de datos.

Nomenclatura	Descripción	Opciones
W	Identificador de include	K
X	Identificador del tipo de include	R: Registro O: Otras C: Comunes E: Excepciones
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas SA: Sanciones VE : Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos
ZZZZ	Identificador del include de 4 posiciones, a elección por la aplicación.	

Tabla 2. Nomenclatura de includes.

Nomenclatura para includes de definición de parámetros de un módulo.

Se nombrarán igual que el módulo al que pertenecen los parámetros definidos en el include, excepto la primera letra, que se cambiará por una "K".

Ejemplo:

Módulo CMHFCFM → Include KMHFCFM.

2.3.3 Ficheros

Los ficheros deberán nombrarse de acuerdo a la siguiente nomenclatura de 7 caracteres:

WXyyzzzz

Nomenclatura	Descripción	Opciones
W	Identificador de fichero	F
X	Identificador del tipo de acceso al fichero	I: Entrada O: Salida A: Entrada/Salida
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas SA: Sanciones VE : Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos
ZZZ	Identificador del fichero de 3 posiciones, a elección por la aplicación.	

Tabla 3. Nomenclatura de ficheros.

2.3.4 JCLs

Los jcl's deberán nombrarse de acuerdo a la siguiente nomenclatura de 7 caracteres:

Wyyzzzz

Nomenclatura	Descripción	Opciones
W	Identificador de tarea	T
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas

		SA: Sanciones VE : Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos
ZZZZ	Identificador del jcl de 4 posiciones, a elección por la aplicación.	

Tabla 4. Nomenclatura de jcl's.

2.3.5 Procedimientos

Los procedimientos que vayan a ser lanzados por el operador desde consola de acuerdo a una planificación deberán nombrarse de acuerdo a la siguiente nomenclatura de 7 caracteres:

YYPzzzz

Nomenclatura	Descripción	Opciones
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas SA: Sanciones VE : Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos
P	Periodicidad de ejecución.	D: Diario S: Semanal M: Mensual A: Anual
ZZZZ	Identificador del procedimiento de 4 posiciones, a elección por la aplicación.	

Tabla 5. Nomenclatura de procedimientos lanzados desde consola.

Los procedimientos que son invocados desde un jcl y no van a ser lanzados desde consola de forma planificada por el operador deberán nombrarse de acuerdo a la siguiente nomenclatura de 7 caracteres:

WYYzzzz

Nomenclatura	Descripción	Opciones
W	Identificador de procedimiento	P
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas SA: Sanciones VE : Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos
ZZZZ	Identificador del procedimiento de 4 posiciones, coincidirá con el mismo valor que el jcl desde el cual es invocado.	

Tabla 6. Nomenclatura de procedimientos invocados desde jcl.

2.3.6 Sysin

La sysin que sea utilizada en procedimientos que vayan a ser lanzados por el operador desde consola deberán nombrarse de acuerdo a la siguiente nomenclatura de 8 caracteres:

YYzzzznn

Nomenclatura	Descripción	Opciones
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas SA: Sanciones VE : Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos
ZZZZ	Identificador de la sysin 4 posiciones, coincidirá con el mismo valor del procedimiento donde es utilizada esta sysin.	
N	Número del 0 al 99.	

Tabla 7. Nomenclatura de sysin utilizadas en procedimientos lanzados desde consola.

La sysin que sea utilizada en procedimientos y jcl's que no van a ser lanzados por el operador desde consola de forma planificada deberán nombrarse de acuerdo a la siguiente nomenclatura de 8 caracteres:

WYYzzzzn

Nomenclatura	Descripción	Opciones
W	Identificador de sysin	S
YY	Identificador del área de negocio.	CO: Conductores TA: Tasas SA: Sanciones VE : Vehículos EX: Exámenes AE: Auto Escuelas CM: Centros Médicos
ZZZZ	Identificador de la sysin 4 posiciones, coincidirá con el mismo valor del procedimiento o del jcl donde es utilizada esta sysin.	
N	Número del 0 al 9 y si es necesario letras.	

Tabla 8. Nomenclatura de sysin utilizadas en procedimientos y jcl's no lanzados desde consola.

2.4 Plantillas de Codificación

Nombre	Descripción
INXXXXX	Programa Online de Negocio
IDATNDB	Programa Online que accede a Base de Datos Relacional (RDB)
IDATRDB	Programa Online que accede a Base de Datos Jerárquica (NDB)
KDATNDB	Include con la definición necesaria para llamar al módulo IDATNDB

Tabla 9. Plantillas de Codificación.

Estas plantillas se han confeccionado con el fin facilitar el mantenimiento de los programas, evitar repetición de código.., entre otras razones. Es posible que a la hora de implementar una funcionalidad se necesite en un mismo módulo utilizar parte de cada una las plantillas de acceso a

datos, esto no es problema, siempre que al hacerlo se mantenga una estructura del módulo clara y similar a la de las plantillas.

En estas plantillas se han incluido DCL de muchas funciones BUILTIN, se deben dejar en el módulo a implementar aquellas que se necesiten.

Los mensajes a visualizar en caso de tener activado el nivel de debug pueden ser más o menos que los que aparecen en las plantillas, pero el número de ellos nunca debe superar en la realidad ni en apariencia al número de sentencias.

2.5 Criterios de nomenclatura

Los nombres de las variables, constantes, etc. deberán de nombrarse de manera lógica con el uso que de ellas se hace, usando un prefijo para identificar el tipo de información que almacena, el nombre debe reflejar el contenido que va a tener.

Tipo	Prefijo
Literales constantes	LT_
Variables	WS_
Fechas	FX_
Horas	HX_
Switches	SW_
Contadores numéricos	CN_

Tabla 10. Nomenclatura general.

3 Reglas de Codificación

A continuación, se exponen las normas de codificación que deberán cumplir los proyectos desarrollados para la Dirección General de Tráfico. Estas normas de codificación serán chequeadas.

Las reglas de codificación tendrán que ser necesariamente cumplidas en todos los casos, no admitiéndose ninguna infracción salvo permiso expreso de la DGT.

Descripción
Los programas de Negocio no pueden tener accesos a Bases de Datos.
No debe codificarse más de una instrucción por línea de escritura.
Utilizar el INCLUDE para estructura de datos comunes de la aplicación.
No utilizar el INCLUDE para codificar funcionalidades, para ello se utilizarán módulos llamados desde donde se necesiten.
El nombre de los procedimientos y funciones deben ser claros y explícitos de acuerdo a su funcionalidad.
Las variables deben definirse explícitamente en una proposición DECLARE o DCL.
Los valores constantes deben estar previamente definidos en variables.
No mezclar tipo de datos diferentes entre operaciones con variables.
Las variables de tipo Numérico, no deben definirse como PIC. Las variables de tipo PIC sólo se han de utilizar para edición, nunca para operar con ellas.
El uso de la instrucción GOTO debe ir ligado exclusivamente a control de base de datos.
En los UNLOAD de JCL BATCH se realizarán las descargas sin formato.
Verificar los códigos de retorno al realizar cualquier tipo de acceso a Base de datos. (DBSCB – DBECB para la BD Jerárquica y SQLCODE para la BD Relacional).
Evitar los IF's anidados. Para hacer comparaciones con más de dos valores, usar siempre la sentencia SELECT.
La sentencia SELECT siempre debe tener codificado la cláusula OTHER.
No se puede utilizar SQL dinámico.
Nunca se deben hacer COMMIT explícitos, ya que el entorno transaccional del Host se encarga de ello.
En el proceso de compilación para pasar a producción se debe utilizar el parámetro OPTIMZE.

Tabla 11. Reglas de codificación.

4 Documentación del Código

El programa fuente generado deberá estar debidamente documentado mediante comentarios dentro del código. Estos comentarios deben ser adicionales al RSA, para realizar matices más técnicos.

4.1 Comentarios en el encabezado del programa

Nombre del programa

Aplicación

Descripción corta de su propósito

```
/*=================================================================*/
/* PROGRAMA:   INEJEMP                                             */
/* APLICACIÓN: SISTEMA DE INFORMACION                              */
/* DESCRIPCION: VALIDACION, MODIFICACION E INSERCION DE            */
/*              DIRECCIONES DE CONDUCTORES                         */
/*=================================================================*/
```

4.2 Comentarios en el encabezado de includes ligados a módulos

Nombre del include

Aplicación

Módulo al que pertenece

```
 /*=================================================================*/
 /* INCLUDE    : KNEJEMP                                            */
 /* APLICACION : SISTEMA DE INFORMACION                             */
 /* MODULO     : INEJEMP                                            */
 /*=================================================================*/
```

4.3 Comentarios en procedimientos

Cada procedimiento o párrafo deberá contener una descripción corta del funcionamiento del código (no más de 5 líneas).

```
/*-----------------------------------------------------------------*/
/* ACTUALIZA EL SET DOMICI DEL OWNER RPER A PARTIR DE LOS DATOS DE  */
/* ENTRADA                                                         */
/*-----------------------------------------------------------------*/

ACTUALIZA_DOMICILIO_RPER: PROC;
...
...
```

5 Normas de Estilo SQL

El presente capítulo muestra las diversas normas aplicables, tanto para la escritura de sentencias del lenguaje SQL como recomendaciones importantes de uso de sus diferentes posibilidades y cláusulas para su uso con la Base de Datos Relacional.

En caso de ser necesario, se seguirán las siguientes recomendaciones a la hora de escribir sentencias SQL. Las presentes normas de estilo serán auditadas por el departamento de calidad.

- ✓ **Referencias a Columnas:** No usar la notación SELECT * y listar cada columna explícitamente, a fin de independizar la sentencia de cambios en la estructura de la tabla.
- ✓ **Expresiones complejas:** Se agruparán las expresiones complejas mediante el uso de paréntesis, incluso cuando la sintaxis del lenguaje no lo requiera a fin de facilitar su entendimiento y evitar errores. En especial, especificar mediante el uso de paréntesis la precedencia de operaciones las aritméticas (suma, resta, multiplicación y división) y de los operadores lógicos (AND, OR y NOT)
- ✓ **Visualización de errores:** Realizar los cálculos de la cláusula WHERE con constantes en lugar de utilizar columnas, siempre que sea posible, para mejorar el rendimiento de la consulta. La conveniencia de hacerlo así radica en que asociando el cálculo a la columna de la WHERE se desactivaría un posible índice asociado a la columna.
- ✓ **Uso de SELECT:** No se harán SELECT de comprobación de existencia antes de hacer INSERT de una fila en una tabla. INSERT ya realiza estas comprobaciones.
- ✓ **Visualización de errores**: En los errores producidos en la parte relacional deben visualizarse los valores de SQLMSG y SQLSTATE.
- ✓ **Uso de NOT**: La cláusula NOT no se debe utilizar nunca en la clausula WHERE.

6 Recomendaciones de optimización

A continuación se describen un conjunto de recomendaciones a efectos de clarificar la codificación.

✓ **Creación de definiciones similares**

En caso de necesitar varias definiciones de datos similares es más sencillo y fiable utilizar la cláusula LIKE para definirlas.

```
DCL 1 ESTRUCTURA,
      3 DATOS....
      3 DATOS...
      3 DATOS....

DCL 1 ESTRUCTURA_1 LIKE ESTRUCTURA;
```

De este modo la definición de ESTRUCTURA_1 será igual que la definición de ESTRUCTURA.

✓ **Comentarios de cierre**

Al cerrar estructuras lógicas (IF, DO, SELECT, etc.) escribir un comentario junto al "END" correspondiente para indicar cuál es su cierre.

```
DO I = 1 TO HBOUND(ENT_DATOS_T_2,1)
    IF WS_PERMI_2(CN_IDX) = WS_CLASE_PERM
      THEN DO;
       SELECT (CN_IDX);
        WHEN (1,2,3)
          DO;
            WS_GRUPO_PERM = '';
          END;
        WHEN (4,5,6,7,8)
          DO;
            WS_GRUPO_PERM = 'G1';
            WS_CLASE_PERM_ANT = WS_PERMI_1(I);
          END;
        OTHER
           DO;
             WS_EDAD_MIN = WS_EDAD_2(CN_IDX);
           END;
      END;                                  /* CIERRA SELECT (CN_IDX) */
   END;                                     /* CIERRA IF */
END;                                        /* CIERRA DO */
```

✓ **Funciones MULTIPLY y DIVIDE (PL/1)**

Utilizar las funciones MULTIPLY y DIVIDE para realizar operaciones. Ello nos evitara problemas de precisión y "overflows".

```
No usar:     TOTAL = SUMA1 / SUMA2
             TOTAL = SUMA1 * SUMA2

Utilizar:    TOTAL = DIVIDE(SUMA1,SUMA2,15,2)
             TOTAL = MULTIPLY(SUMA1,SUMA2,15,2)
```

- ✓ **Utilización de paréntesis**

Se utilizarán paréntesis para AND y OR, procurando que los AND estén fuera y los OR dentro. Ejemplo: (A=B OR C=D) AND (E=F OR G=T).

- ✓ **Cláusula WHERE**

En la cláusula WHERE se accederá siempre por el índice o parte de él y poner éste en la parte más próxima al WHERE. Si es por parte, procurar que la parte conocida sea la primera.

- ✓ **Consultas On line**

En On line, al recuperar varias filas, se debe conocer el número de filas recuperadas, para poder acotar la consulta en el caso de que sean muchas.

- ✓ **Priorización de operadores**

Los operadores que se utilicen en la cláusula WHERE deben utilizarse priorizando los de menor valor en la columna prioridad de la siguiente tabla:

Prioridad	Operador	Ejemplo
1	=	A = 2
1	IN	A IN (‘P’, ‘Q’, ‘R’)
2	BETWEEN	A BETWEEN 1 AND 100
2	LIKE (prefijo)	A LIKE ‘RDB%’
3	LIKE (intermedio/sufijo)	A LIKE ‘%RDB’
4	>, <; >=, <=	A >=100
5	Predicado cualificado	A > ANY (SELECT …)

Tabla 12. Priorización de operadores.