Lenguajes de programación COBOL y PL/I, historia y características principales

Rolando Lázcares Valenciano
Universidad de Costa Rica,
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica
rolandolv@gmail.com

Álvaro Molina García
Universidad de Costa Rica,
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica
varo87@gmail.com

Resumen

La historia del lenguaje de programación PL/I data desde 1960 cuando un comité liderado por IBM empezó a desarrollar el proyecto. El lenguaje tuvo muchos aspectos innovadores en cuanto a la programación de la época, pero a pesar de esto tuvo muchos problemas de implementación y de aceptación con los clientes y los programadores. Por otra parte el lenguaje de programación COBOL consta de su propia historia la cual toma lugar desde 1959 y nace como herramienta clave para el manejo estructurado de datos en cuanto a gestión de negocios se trata. Como bien se sabe, todo bien tiene su contraparte y los mismos beneficios de COBOL se convierten en sus defectos, ya que su enfoque específico y su poca flexibilidad le encapsulan y no le permiten abrirse a otras áreas.

1 Introducción

El lenguaje de programación PL/I, nació entre las décadas de los 60 y 70, y vino a funcionar dos grandes grupos que había en la época en programación. A pesar de esto el lenguaje no fue muy bien recibido en el mercado porque debido a errores de diseño, tuvo muchos problemas de implementación, y no fue del agrado de los clientes y programadores.
Mientras se desarrollaba PL/I también en la década de los 60 se comenzó un proyecto de un lenguaje de programación el cual funcionaría en toda plataforma y todo tipo de máquina, este proyecto fue llamado COBOL y se creó con el fin de manejar toda gestión de negocios posible, independientemente de la empresa y negocio asociado del que se tratara.
Ambos lenguajes cuentan con diversos temas cruciales para un mayor entendimiento acerca de estos, a continuación dichos temas serán tratados individualmente y por aparte para cada lenguaje.

2 Historia Lenguaje PL/I

El lenguaje de programación PL/I (programming languange I), fue creado alrededor de las décadas de los 60 y 70, por IBM. Nació como parte del desarrollo de la arquitectura system 360, fabricada por IBM en aquel tiempo [1].
Su desarrollo fue llevado a cabo en los laboratorios hursley, ubicados en el reino unido y propiedad de IBM, y su diseño iba orientado a satisfacer las necesidades existentes en las aplicaciones científicas y comerciales [2].
Esta necesidad, era producida debido a que los lenguajes de programación antecesores a este, como por ejemplo Cobol y Fortran, podían resolver problemas de carácter científico o comercial, pero era necesario realizar aplicaciones que pudieran resolver tanto los problemas de carácter científico, como los referentes al negocio, y dado esto se procedió a realizar este lenguaje [2].
Originalmente, el lenguaje tuvo el nombre de NPL (new programmin language), pero en el reino unido ese nombre ya estaba tomado por el national phisical laboratory, por lo que tuvieron que buscar en nuevo nombre, y antes de llegar al definitivo, nombres como MPL y MPPL se tuvieron a consideración [2].
Dentro del equipo de trabajo de la elaboración del lenguaje, se encontraban un comité de programadores de la empresa IBM, y un grupo de usuarios, necesitados del software, de todo los estados unidos. El proyecto se trabajó durante meses [2].
Otro tema importante a tratar es el lenguaje en sí y será analizado en el siguiente tema.

3 Lenguaje PL/I

PL/I es un lenguaje estructurado por capas de tercera generación, donde la capa externa corresponde a la parte del código y datos, y posee distintas capas que corresponden a procedimientos que son ejecutados como subrutinas o como funciones, y otras capas iniciales, que poseen llamados a funciones in-line [4].
Como dato importante de este lenguaje, podemos mencionar que carece de palabras reservadas. Posee una serie de palabras reservadas pero que son reconocidas únicamente por contexto, de otra manera el programador puede utilizarlas libremente, ya que el compilador puede notar la diferencia según el contexto en el que se utilicen estas palabras [5].
Otro punto importante que es bien implementado por este lenguaje, es el control adecuado de y completo sobre las variables que se declaren en un programa específico. El lenguaje permite que al declarar una variable, se pueda especificar el tamaño de esta en cuanto a longitud e inclusive en cuanto a la cantidad de bits que gastará de espacio en memoria, y permite hacer revisiones para garantizar que estos parámetros de espacio definidos no sean violados.
PL/I posee una gran diversidad en tipos de variables, cuyas palabras reservadas, que anteriormente definimos que se emplean según el contexto, están denotadas en inglés, entre ellos podemos mencionar los tipos aritméticos: FIXED, FLOAT, REAL o COMPLEX. Entre otros tipos de variables, se encuentran POINTER, OFFSET, GRAPHIC, PICTURE, LABEL, ENTRY, CHARACTER, BIT, AREA, EVENT, FILE, entre otras más [4].
Como otro de los aspectos principales de este lenguaje, podemos mencionar que incluye muchos complementos a la programación por defecto. Esto se ha desarrollado con el objetivo de aligerar la carga que posee el programador. De esta manera, el programador solo tendrá que proveer la información mínima, y el compilador podrá ser capaz de incluir el resto por defecto [4].
También posee un gran número de almacenamiento de atributos. Los valores por defecto son almacenados de manera automática, al inicio de una capa, con su respectiva inicialización y las variables son liberadas al terminar la capa. Las variables estáticas son almacenadas durante toda la vida útil del programa, y otro tipo de almacenamiento, como el controlado debe de estar especificado en el programa, al igual que una pila. El almacenamiento externo se comporta casi que igual al estático, con la diferencia que puede ser referenciado por programas externos.
Otro dato importante es que las funciones de entrada y salida (E/S) en PL/I vienen definidas dentro del lenguaje, y no son funciones que sirven de complemento al lenguaje a través de bibliotecas. El lenguaje permite que los

archivos puedan ser procesados como cadenas de bits o como registros individuales, con longitud variable o fija. También, estos archivos pueden ser accesados de manera secuencial o aleatoria [4].
Entre otro de los aspectos que incluye el lenguaje se encuentra el detallado manejo de excepciones, que pueden ocurrir de 3 formas: asíncronos mediante errores de E/S, de hardware por problemas como el desbordamiento o generadas por el programa. Para cada tipo, se pueden declarar manejadores que pueden ser almacenados en una pila y sacarlos de la pila, y algunas de estos pueden estar habilitados o deshabilitados por el programa según el nivel de la capa [4].
Además para especificar mejor este lenguaje se describirán a continuación las diferentes características principales.

4 Características Principales de PL/I

A continuación se presentarán una serie de aspectos que consideramos son los más relevantes que posee el lenguaje PL/I.

1. El compilador hace uso de palabras reservadas analizándolas según el contexto, lo que brinda mayor libertad al programador.
2. Presenta un alto porcentaje de modularidad, gracias a su estructura por capas.
3. Presenta cuatro distintas formas de almacenamiento: AUTOMATIC, STATIC, CONTROLLED, y BASED, lo que le brinda mayores opciones al programador.
4. Entre sus definiciones, esta la utilización de estructuras de datos, tales como vectores, estructuras, vectores de estructuras, etc.
5. La definición de los datos a utilizar en un programa es independiente al hardware de la máquina.
6. Funciones de entrada y salida forman una parte íntegra del lenguaje, y no a través de alguna biblioteca.
7. Pertenece al software libre, por lo que es gratuito.
8. Permite realizar aplicaciones que resuelvan todo tipo de problemas (científicos y comerciales) [3].

Para ver mejor de que se está hablando, se presentarán varios ejemplos.

5 Ejemplos de PL/I

A continuación se presentarán una serie de ejemplos de programas escritos en PL/I.
5.1 Hola Mundo.

```
Mundo: Procedure options(main);
Put List( 'Hola Mundo' );
End Mundo; [3]
```

5.2 Búsqueda de una hilera.

```
BUSCARHILERA: PROCEDURE OPTIONS(MAIN)

DECLARE PAT VARYING CHARACTER(100),
        LINEBUF VARYING CHARACTER(100),
        (LINENO, NDFILE, IX) FIXED BINARY;

 NDFILE = 0; ON ENDFILE(SYSIN) NDFILE=1;
 GET EDIT(PAT) (A);
 LINENO = 1;
 DO WHILE (NDFILE=0);
   GET EDIT(LINEBUF) (A);
   IF LENGTH(LINEBUF) > 0 THEN DO;
     IX = INDEX(LINEBUF, PAT);
     IF IX > 0 THEN DO;
       PUT SKIP EDIT (LINENO,LINEBUF)(F(2),A)
     END;
   END;
   LINENO = LINENO + 1;
 END;
 END BUSCARHILERA; [2]
```

Cada lenguaje tiene sus defectos y así se verá en la siguiente sección para este lenguaje.

6 Problemas asociados a PL/I

A pesar de que el resultado final obtenido una vez concluido el proyecto de la construcción de este lenguaje resultó en un lenguaje fácil de aprender y de utilizar, no tuvo los resultados comerciales esperados.
Una de las razones por la que tuvo malos resultados comercialmente, fuel por el alto costo en tiempo y la alta dificultad al desarrollar compiladores eficientes para el lenguaje. Esto surge como causa de la organización a la hora de desarrollar el software, en cuanto a realizarlo de tal manera que cumpliera los requisitos de construir aplicaciones científicas y comerciales, y de dejarle el desarrollo del producto a un comité, organizado por clientes y programadores. Otro de los aspectos que hizo difícil la implementación de compiladores, que otros lenguajes de alto nivel como Cobol si implementaban, fue la falta de niveles pequeños o subniveles, y esto ocasionaba la aplicación de cada elemento del lenguaje para la correcta satisfacción de las normas definidas.
Otro problema que tenía el PL/I fue el de las palabras reservadas. Recordemos que a este lenguaje no se le establecieron palabras reservadas, sino que se obtenían según el contexto, y esto añadía mucha dificultad a la hora de analizar el código y determinar de cuando se refería a un nombre de variable y cuando a algún tipo predeterminado [4].
También se hizo presente un problema de política. Al ser el lenguaje implementado por IBM, los competidores no les atraían mucho la idea de invertir en un lenguaje diseñado por IBM, en especial porque la inversión debía de incluir los gastos de recuperación de información, proveído por un comité de lenguajes de IBM.
Todos estos problemas surgieron debido a la falta de importancia que se le dio al organizar el desarrollo del lenguaje. La complejidad de diseñar un compilador no fue un tema que se consideró al inicio del proyecto. La ambición de los diseñadores, de querer realizar un lenguaje que cumpliera con las necesidades de todos sus clientes. Además la falta de palabras reservadas contrajo una serie de problemas que dificultó el análisis del código [4].
A la vista de los programadores, el lenguaje no presentaba las suficientes comodidades como para hacer el cambio de cobol o fortran hacia este lenguaje. La falta de versiones eficientes de compiladores, su complejidad en cuanto a la programación multihilo, le ponía trabajo extra al sistema operativo. También, los programadores de esa época estaban fuertemente divididos en dos grupos, los científicos informáticos y los programadores de negocios, que utilizaban Fortran y Cobol respectivamente. Pl/I había sido diseñado especialmente para hacer la gran unión entre estos dos grupos, y por ello tenía entre su sintaxis, fragmentos de las sintaxis de ambos lenguajes. Sin embargo, una gran rivalidad separaba a estos dos grupos, que en cuanto vieron a este nuevo lenguaje con extractos de sus rivales, adoptaron una actitud de rechazo ante este nuevo lenguaje [4].
Todos estos problemas, hicieron de PL/I un lenguaje con poco éxito comercial, al no contar con el apoyo ni de los programadores de la época ni de las empresas. A pesar de todo esto, se dice que sin la producción de este lenguaje, el equipo del US Project Apollo no hubiera podido lograr que un hombre llegara a la luna.
A continuación se analizará el siguiente lenguaje de programación en mención.

7 Historia de COBOL

Para explicar de la mejor manera como COBOL (COmmon Business Oriented Language) por sus siglas en inglés, dio inicio en el mundo de la informática es necesario regresar al pasado y profundizar el contexto en el cual fue desarrollado, esto data de aproximadamente un poco mas de 35 años. Cada computadora traía su propio sistema operativo y además su propia programación.
En mayo de 1959, en Estados Unidos, se creó una comisión denominada CODASYL (Conference On Data Systems Languages) por sus siglas en inglés, estaba integrada por una serie de representantes de todo tipo: fabricantes de computadoras, empresas privadas y representantes del gobierno, su propósito fue el deseo de desarrollar un lenguaje que fuera aceptado por toda marca de computadora. Así se creó COBOL, un lenguaje completamente orientado a negocios y el cual en su primera versión fue llamado COBOL-60 por su año de lanzamiento.
Luego conforme fueron pasando los años y el manejo de COBOL era mucho más regular, surgieron recomendaciones tanto de usuarios normales como de expertos en la materia, dando lugar a varias revisiones en distintos años: 1961, 1963, 1965. Todas las anteriores dieron paso a la primera versión Standard la cual nació en 1968, a su vez esta fue revisada en 1974 y fue denominada COBOL ANSI o COBOL-74. La última versión revisada es la de 1985 llamada COBOL-85. [8]
Como su nombre lo dice COBOL es un lenguaje orientado a gestión de negocios, y no es desconocido para nadie que toda empresa, independientemente de la actividad a la cual se dediquen, tiene un sistema para gestionar su tipo de negocio, por lo tanto COBOL pretende brindar mayores facilidades para la creación de programas que le permitan a dichas empresas gestionar específicamente su negocio.
Seguidamente se hablará del lenguaje COBOL.

8 Lenguaje COBOL

COBOL es un lenguaje compilado de primer nivel. Este consta de un código fuente legible y el cual se basa en una serie de reglas. Al igual que los diversos lenguajes de su generación puede ser escrito en cualquier editor de texto. Genera un código objeto (ya compilado) el cual esta listo para ser ejecutado.

8.1 Estructura del lenguaje

Este lenguaje consta de cuatro partes:

8.1.1 La División de Identificación

La División de Identificación tiene la siguiente estructura:

```
IDENTIFICATION DIVISION
PROGRAM-ID. NombreDelPrograma.
[AUTHOR. NombreDelAutor.]
Otras entradas
```

La primera línea representa la cabecera de la división y aquí es donde comienza el programa. Luego viene el nombre del programa el cual es definido por el programador, también se especifica el nombre del autor y otras entradas de las cuales muchas son simplemente información para el programador (comentarios).

8.1.2 La División del Entorno

Esta sección especifica las características físicas del ambiente en el que correrá el programa. Muestra la información de la computadora en la cual se generó el programa y en la cual se va a correr. Además es la encargada de relacionar los dispositivos de entrada y salida del programa con su respectivo hardware.
Todo esto se realiza con el fin de facilitar la modificación del programa cuando este se pretende ejecutar en otra máquina, o cuando se ejecuta con varios dispositivos periféricos.
Existen varios detalles más que deben ser especificados aquí, como las secuencias de comparación, el símbolo de moneda, o el símbolo de la coma decimal.

8.1.3 La División de Datos

Contiene los nombres de los datos que serán procesados por el programa COBOL. Este puede constar de varias secciones, las dos principales son la de Archivos, en la cual se describen los datos que se envían o reciben de los periféricos de la máquina, y la de Trabajo/Almacenamiento en la cual se describen las variables del programa. Se pueden especificar otras secciones, como la de Linkado, utilizada para subprogramas, y la de Informes, para programas generadores de informes.

8.1.4 La División de Procedimientos

Esta es jerárquica y consta de los procedimientos, secciones, párrafos, declaraciones, sentencias e instrucciones necesarias para ejecutar el procesamiento de una función y a su vez procesar los datos inherentes a esta. Aquí es donde el programador genera sus algoritmos de manejo de datos. [9]

8.2 Características Generales de COBOL

COBOL es un lenguaje independiente de la plataforma donde se ejecute, además puede comunicarse con cualquier base de datos que exista, se adapta a la tecnología cliente-servidor, a la tecnología de eventos e inclusive puede estar en la web.
Posee un elevado grado de precisión y velocidad de cálculo numérico, con la capacidad de manejar hasta 30 posiciones decimales. [6]
Se emplea en aplicaciones comerciales y para el manejo de grandes cantidades de datos. Como dato histórico es el lenguaje mas utilizado en toda la historia para este propósito.
COBOL cuenta con aproximadamente 300 palabras reservadas.
COBOL tiene características principales únicas las cuales serán analizadas a continuación.

9 Características principales de COBOL

A continuación se presentarán una serie de características principales del lenguaje COBOL:

1. Es un lenguaje auto-documentado: se suponía en un inicio que COBOL sería un lenguaje accesible para agentes no programadores, o sea se hablaba de que estos pudieran revisar el código sin tener conocimientos de programación, la idea a pesar de que no fue completamente efectiva, hizo que cobol se convirtiera en el lenguaje auto-documentado más fácil de entender.

2. Es un lenguaje simple con una funcionalidad limitada: no tiene punteros ni funciones ni tipos definidos por el usuario.

3. Es portable: su estándar no pertenece a ninguna marca concreta y puede ser llevado a todo tipo de máquinas por ejemplo Windows, UNIX, OS/2 entre otros.

4. Es mantenible: como se ha dicho ya, COBOL tiene una gran facilidad de interpretación y una gran legibilidad y además tiene una rígida estructura jerárquica y por medio de todo esto su mantenimiento se facilita muchísimo. Como ejemplo se podría ver que en el año 2000 con el problema Y2K muchas aplicaciones COBOL se vieron afectadas y de igual manera su reparación fue mucho más fácil y barato en comparación con otros lenguajes.

Seguidamente se ejemplificará lo hablado anteriormente acerca de COBOL.

10 Ejemplos COBOL

10.1 Multiplicación de dos números

```
IDENTIFICATION DIVISION.
PROGRAM-ID. Secuenciacion.
AUTHOR. Carlos S Melon Fdez.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 Num1 PIC 9 VALUE ZEROS.
01 Num2 PIC 9 VALUE ZEROS.
01 Resultado PIC 99 VALUE ZEROS.
PROCEDURE DIVISION.
CalculaResultado.
ACCEPT Num1.
ACCEPT Num2.
MULTIPLY Num1 BY Num2 GIVING Resultado.
DISPLAY “El resultado es = ”, Resultado.
STOP RUN.
```

10.2 Programa que maneja condiciones

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  Iteration-If.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01  Num1           PIC 9  VALUE ZEROS.
01  Num2           PIC 9  VALUE ZEROS.
01  Result         PIC 99 VALUE ZEROS.
01  Operator       PIC X  VALUE SPACE.

PROCEDURE DIVISION.
Calculator.
    PERFORM 3 TIMES
       DISPLAY "Enter First Number      : " WITH NO ADVANCING
       ACCEPT Num1
```

```
      DISPLAY "Enter Second Number     : " WITH NO ADVANCING
      ACCEPT Num2
      DISPLAY "Enter operator (+ or *) : " WITH NO ADVANCING
      ACCEPT Operator
      IF Operator = "+" THEN
         ADD Num1, Num2 GIVING Result
      END-IF
      IF Operator = "*" THEN
         MULTIPLY Num1 BY Num2 GIVING Result
      END-IF
      DISPLAY "Result is = ", Result
   END-PERFORM.
   STOP RUN.
```
[10]

COBOL tiene sus propias limitaciones, estas se verán en la siguiente sección.

11 Problemas asociados a COBOL

En contraste con otros lenguajes de programación, Cobol no fue concebido para cálculos complejos matemáticos o científicos, de hecho solo dispone de comandos para realizar los cálculos más elementales.
Es un lenguaje muy antiguo y consecuentemente sus peculiaridades extra pueden llegar a ser muy irritantes para programadores acostumbrados a otros lenguajes. [9]
Las versiones modernas de COBOL han introducido una serie de mejoras para generar programas bien estructurados, sin embargo esto se ve afectado pues COBOL cuenta con una serie de elementos los cuales al ser utilizados dificultan e incluso imposibilitan la creación de un programa bien estructurado.
Entre las quejas más comunes de los usuarios de COBOL están:

- Falta de funcionalidad
- Poca Flexibilidad
- Solamente un enfoque al cual tratar (Gestión de negocios)

12 Conclusiones

Se puede mencionar que el lenguaje PL/I implementa muchas características que demostraron gran avance en la programación en la década de los 70. Logró dar el paso para formar un lenguaje de alto nivel más íntegro, que pudiera cumplir con las necesidades de una gran variedad de clientes.

PL/I introdujo términos de programación que eran nuevos en la época, y que impulsaron a lo que podemos considerar como un nuevo modelo de programación, mediante la implementación de estructuras de datos, multihilo, punteros y muchas otros aspectos más que lenguajes como C y C++ emplearon.

A pesar de todos estos aspectos innovadores que incluía el lenguaje, no contó con el apoyo necesario para poder comercializarse de la manera esperada. En especial por el poco agrado de los programadores respecto al producto y el poco apoyo de las empresas compradoras.

Se puede concluir PL/I abrió la puerta a nuevos e innovadores lenguajes de programación, y que sirvió como conejillo de indias ante el miedo al cambio, y ante los errores que este nuevo modelo podía ocasionar. Sirvió a otros lenguajes para poder implementar estas ideas nuevas en la programación pero de una manera más atractiva para programadores y clientes.

COBOL es un lenguaje de primer nivel cuya aplicación está directamente asociada con la gestión de negocios y todo lo relacionado a esto.

El lenguaje COBOL es capaz de manejar grandes cantidades de datos y manipularlos para obtener excelentes resultados.

Siendo COBOL un lenguaje tan antiguo y el cual todavía se utiliza en grandes empresas, da mucho de que hablar pues su funcionalidad específica a pesar de ser anticuada es efectiva y genera los resultados deseados por quienes lo utilizan.

Lastimosamente COBOL al solo tener un enfoque no cuenta con capacidades más allá de este, dejando poco espacio para su utilización en otros tipos de problemas.

Referencias

[1] http://sunblade.iespana.es/Informatica/Lenguajes/Informatica-Lenguajes.html

[2] http://en.wikipedia.org/wiki/PL/I

[3] http://www.engin.umd.umich.edu/CIS/course.des/cis400/pl1/pl1.html

[4] http://home.nycap.rr.com/pflass/pli.htm

[5] Sprowls, R. Clay.” Introduccion a la programacion en lenguaje PL/I”. Harper and Row. New York, N.Y, Estados Unidos. 1971.

[6] McCracken, Daniel D. “COBOL programación estructurada”. Primera edición. Limusa. México, 1984.

[7] Newcomer, Lawrence R. “PROGRAMACION (COMPUTADORAS ELECTRONICAS) COBOL”. Primera edición. McGraw-Hill. México, 1986.

[8] http://www.escobol.com/modules.php?name=Sections&op=viewarticle&artid=39

[9] http://webs.enterate.com.ar/Web/Computacion/mrebollo/Progra3/Estructura_de_un_programa_COBOL.PDF

[10] http://www.csis.ul.ie/COBOL/examples/conditn/IterIf.htm