Historia de los Lenguajes de Programación

AÑOS 1960-1969

MANUEL ÁNGEL RUBIO JIMÉNEZ

Historia de los Lenguajes de Programación

Años 1960-1969

Manuel Ángel Rubio Jiménez

ISBN 978-84-124520-4-4

Capítulo 1. RPG y FARGO

> En el principio, Dios creó el cielo y la tierra. Luego creó RPG para ordenarlos.
>
> — Anónimo

Estos lenguajes desarrollados en 1959 dentro de IBM se encargaban de realizar una transición entre los sistemas de tabulación a las nuevas computadoras. Para entender mejor porqué estos sistemas de tabulación fueron tan importantes vamos a retroceder un poco en la Historia para entender en qué consistía esta máquina de tabulación y cómo ayudaba este lenguaje a realizar la transición a las nuevas máquinas.

El lenguaje RPG dadas sus siglas significa *Report Program Generator* o Generador de Programas de Informes y fue diseñado principalmente para la transición a máquinas de System/360.

Antes de RPG también existió FARGO, *Fourteen-o-one Automatic Report Generation Operation* u Operación de generación automática de informes 1401, pensado para la máquina IBM 1401. De este lenguaje hablaremos un poco más adelante.

Ahora comencemos por el principio, desde Hollerith.

1.1. Del censo de 1890 a IBM

Al igual que Charles Babbage muchos matemáticos como Herman Hollerith tenían trabajos mecánicos y muy pesados. Mientras que Babbage sabemos que construyó su Máquina Diferencial para ayudarle en las tareas de cálculo, en los Estados Unidos de América Hollerith se inspiró en conductores de los billetes de tren que hacían agujeros en posiciones diferentes para almacenar datos del pasajero.

Hollerith había sufrido la ardua tarea junto con muchos más estadistas de realizar el censo de 1880, aunque él solo estuvo en esa tarea hasta agosto de 1883, el censo no se concluyó hasta 8 años más tarde. Curiosamente entre 1882 y 1883 Hollerith fue también instructor de ingeniería mecánica en el Instituto de Tecnología de Massachusetts (MIT).

Hollerith entró a trabajar en la Oficina de Patentes durante aproximadamente un año y cuando dejó el trabajo publicó el desarrollo de una máquina para tabular estadísticas de población y recibió las patentes sobre esa máquina en 1889.

Este sistema se utilizó para el censo de 1890, el primero que se automatizó en cierto grado.

Al igual que con el telar de Jacquard y las Máquinas de Babbage, el uso de tarjetas perforadas fue el estándar de la época y Hollerith supo aprovechar esta tecnología para realizar las tablas necesarias para el censo de los Estados Unidos.

Las tarjetas disponían la información y según los datos de cada ciudadano se perforaban acumulando millones de tarjetas para ser contabilizadas más tarde por la máquina y obtener los resultados dependiendo de qué se quisiera o necesitara obtener.

Las máquinas funcionaron tan bien que de este trabajo surgió una organización comercial llamada Tabulating Machine Company o Compañía de Maquinaria de Tabulación.

Durante este período se vendieron muchas de estas máquinas no solo para realizar el censo de los ciudadanos sino también otras compañías las adquirieron para realizar facturas y otras muchas aplicaciones dentro del mundo empresarial.

El funcionamiento de estas máquinas consistía en introducir un conjunto de tarjetas perforadas. Estas perforaciones se situaban en un espacio concreto y según ese espacio tenían un significado u otro. Cuando estas tarjetas eran procesadas por la máquina, la máquina registraba el número realizando una suma en alguno de sus acumuladores. En versiones muy antiguas, las sumas se registraban en algún dial de la máquina mientras que las últimas versiones realizaban una impresión del acumulador por cada entrada de datos.

Existían tarjetas maestras con tan solo una perforación en un lugar específico que permitía a la máquina hacer una tarea concreta. Estas se conocían como cartas maestras. Estas podían tener información más completa como la información de un cliente para una factura y esta información era impresa en ese momento para después contabilizar las tarjetas de los artículos adquiridos uno a uno. Cuando se detectaba la siguiente tarjeta maestra la página imprimía el acumulador y expulsaba la hoja.

En 1911 la empresa pasó a llamarse Computer-Tabulating-Recordings Company y en 1914 Thomas J. Watson Senior se unió a la compañía cambiando su nombre definitivamente a International Business Machines Corporation o IBM.

1.2. La migración al IBM 1401

IBM compitió con Remington Rand Corporation durante el inicio del siglo XX. IBM lanzó en 1949 el IBM 407 y Remington Rand el Remington Rand 409. La competición duró hasta la década de 1950 cuando comenzó a surgir la computación y estas máquinas de tabulación fueron desplazadas.

Los computadores en la década de 1950 se distribuían con un lenguaje Autocoder, un ensamblador avanzado y algo pesado por lo que también

incluían el Symbolic Programming System (SPS) también ensamblador pero más ligero que Autocoder. Con estos lenguajes, ¿por qué incluir FARGO en lugar de enseñar a los usuarios a desarrollar en ensamblador?

Cada gran cambio requiere de ayudas y para migrar de un sistema tan mecánico como las máquinas de tabulación a otros basados en programas como los computadores requería de un sistema que simulase la forma de interactuar con la máquina. Cuando IBM lanzó su serie IBM 1400 para sustituir a sus tabuladores IBM 400 agregó el lenguaje de programación FARGO.

De hecho, su nombre indica perfectamente su cometido: *Fourteen-o-one Automatic Report Generation Operation* u Operación de generación automática de informes 1401. Es decir, un programa para generar informes para el IBM 1401.

FARGO simulaba la noción del ciclo de la máquina y el material distribuido por IBM (Corporation, 1964) mostraba las relaciones entre el panel de control y la hoja de codificación del lenguaje de programación. El personal encargado de operar las máquinas de tabulación estaban acostumbrados a cambiar cables en el panel de control para obtener los resultados deseados sumando, restando, multiplicando y dividiendo.

Este personal era el usuario objetivo del lenguaje, los perfectos candidatos para aprender FARGO cambiando esta dinámica de conectar cables por escribir estas órdenes en una hoja de procesamiento.

Rellenando estas hojas se consigue un programa para procesar las tarjetas del mismo modo que funcionaban las máquinas de tabulación. Estas hojas eran necesarias y debían adquirirlas también a IBM al igual que las tarjetas perforadas que aún se seguían usando para la entrada de datos.

Una de las características del lenguaje es la interpretación de la información. No requería de un proceso de compilación solo del procesamiento de la hoja con las instrucciones para generar los datos de salida una vez se ha completado la cantidad de entrada esperada.

El uso de FARGO despuntó durante la explotación y migración a la serie IBM 1400 y aunque se ofrecieron simuladores para seguir utilizando FARGO en la serie System/360, en estos sistemas eran más empleados COBOL, FORTRAN y RPG.

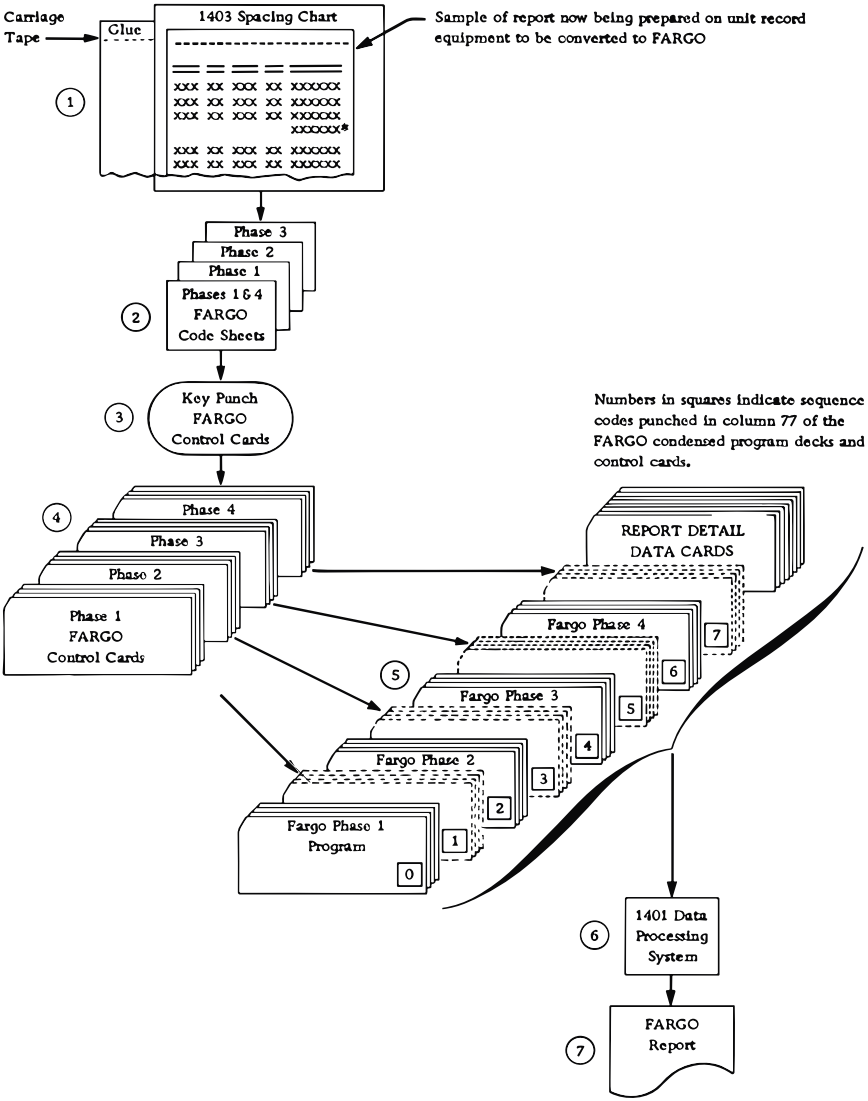

Figura 1. Hoja de Migración a FARGO

No se menciona en ningún documento quien creó FARGO pero veremos más adelante que hay razones para creer que FARGO y RPG fueron el mismo lenguaje pero con un cambio de nombre o un nombre específico por lo que cabe pensar que Wilf Hey fue su creador.

A lo largo del documento de FARGO de IBM (Corporation, 1964) se muestran ejemplos de cómo se escriben los programas en unas hojas formateadas para tal propósito y cómo a través de las fases mencionadas en la Figura 1 se va procesando el programa hasta tener preparada la hoja final que nos servirá para introducir en la máquina, se trata de un procedimiento manual para llevar a cabo lo que buscamos conseguir.

1.3. Impresión y Caracteres

En las primeras impresoras la disposición de los caracteres era muy limitada. No es como con UTF-8 o UTF-16 que disponemos de tablas de caracteres donde entran millones de posibles símbolos. En esta década para poder imprimir cajas, líneas y otros símbolos debían ordenarse a la impresora cambiar la secuencia de clasificación y emplear otro conjunto de caracteres diferente.

Así, cuando se ordena a la impresora que imprima el símbolo que en la tabla ASCII era relativo a la letra A, en otra tabla diferente el carácter que se imprime puede ser la esquina superior izquierda de una tabla para dibujar un marco donde se incluirán los números de los totales.

1.4. System/360 y RPG

El lenguaje creado para servir de migración de las máquinas de tabulación aún existentes a la serie de máquinas System/360 fue RPG por sus siglas *Report Program Generator* o Generador de Programas de Informes. Fue influenciado por el lenguaje FARGO y Wilf Hey fue su creador.

Hay muchos datos confusos con respecto al origen de FARGO y RPG dando a entender que ambos estaban disponibles en la serie de

computadores IBM 1400 mientras que otros documentos argumentan que RPG 0 fue nombrado FARGO por ser específicamente para IBM 1401 mientras que RPG I y sus sucesivas versiones al estar basadas en otras series de computadores obtuvo el nombre genérico RPG.

Las fuentes que citan la creación e inclusión de RPG en la familia de IBM 1400 junto con FARGO aunque no dejan clara su relación indicando que RPG fue influenciado por FARGO y sin embargo, ¿cómo pudo influir FARGO en RPG si ambos sitúan su fecha de creación el mismo año? Si esa influencia es correcta, por fuerza FARGO debía existir antes de que RPG hubiese sido lanzado.

Por lo tanto parece más fehaciente que FARGO fuese la versión RPG 0 del lenguaje RPG y por ello ambos estuviesen disponibles en la serie de IBM 1400 mientras que FARGO tan solo podía correr simulado en la serie System/360 y sus otras versiones RPG I y sucesivas sí que se ejecutaban en System/360 (Kelly, 2009).

Otro de los motivos por los que se realizó un cambio de nombre de FARGO a RPG pudo ser porque la versión RPG 0 en la serie IBM 1400 tenía muchos fallos y por ese motivo se dedicase mayor tiempo al desarrollo de una versión superior y motivase el cambio de nombre.

1.5. ¿Cómo programamos en RPG?

Se dice de RPG que fue una versión mucho mejor que FARGO pero no hay ningún manual o guía de referencia que haya sobrevivido o haya sido almacenada. Quizás no se hizo ninguna al tiempo de su lanzamiento. Sin embargo, para RPG II lanzado a mediados de 1960 sí existe una guía de 1977 de Control Data Corporation (Corporation, 1977).

Este lenguaje es de tamaño fijo, es decir que cada espacio dentro de cada línea tiene un significado específico y no podemos agregar espacios o saltos de línea en cualquier sitio. El ancho de la línea es de 80 caracteres lo cual se corresponde al ancho de las tarjetas perforadas que tenían también 80 posiciones horizontales.

El código comienza reservando las primeras 2 columnas para el número de la página, las siguientes 3 columnas para la línea, la sexta columna para el tipo de hoja:

Cabecera (H)

Proporciona información perteneciente a la compilación.

Fichero (F) y Extensión (E)

Especificaciones de descripción de archivo y extensión. Describen los ficheros, tablas y cadenas de datos (arrays) a ser usados en el programa.

Contador de Líneas (L)

Proporciona información sobre las líneas de fichero impresas producidas por el programa.

Entrada (I) y Salida (O)

Describe los registros en los ficheros nombrados en F y E.

Cálculos (C)

Describe las operaciones a ser realizadas sobre los datos de entrada.

Cada hoja proporcionada al sistema solo puede ser de un tipo. Esto quiere decir que si necesitamos proporcionar información a ser impresa necesitamos una hoja de tipo H donde indiquemos toda la información a ser impresa directamente. Si necesitamos indicar el fichero para concretar el contenido de ese fichero necesitaremos una hoja F, datos de entrada en hojas I, los cálculos a realizar en hojas C y el formato de salida en hojas O.

El programador también puede introducir tablas o cadenas de datos como constantes, tablas de traducción que sí pueden cambiar y tablas de secuencia de clasificación alternativa para alternar el conjunto de caracteres. Esta información se agrega en las hojas H en columnas específicas.

Podemos ver un ejemplo de una hoja para cálculo (C) en la Figura 2 donde se muestran las columnas iniciales reservadas y ya mencionadas anteriormente y las operaciones ocupando los espacios titulados *Factor 1*, *Operation* y *Factor 2*. Podemos encontrar comentarios al final de cada línea.

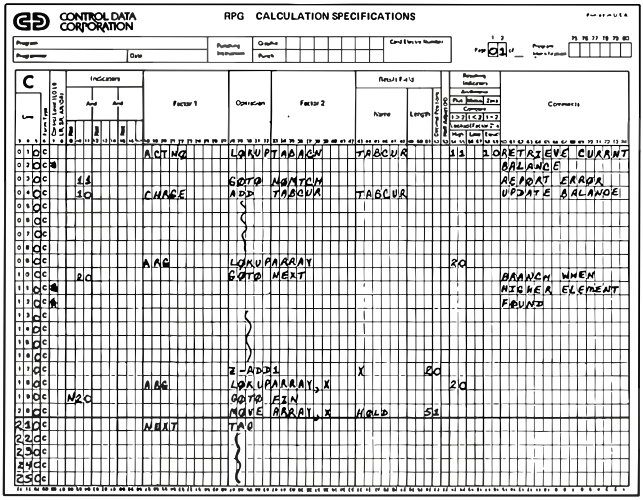

Figura 2. Especificación de Cálculo. Hoja C.

1.6. El presente y futuro de RPG

El lenguaje de programación RPG II fue muy extendido y utilizado durante la década de 1960 y 1970. A finales de 1970 fue lanzado RPG III y en 1994 RPG IV. A principios del siglo XXI RPG tuvo su mayor actualización cambiando la forma en la que se realizaban los programas y sigue a día de hoy siendo actualizado y usado.

No obstante, la restricción de espacio fijo no fue eliminada hasta 2015.

El lenguaje RPG ha estado disponible en diferentes sistemas operativos, no solo propiedad de IBM y hemos podido ver entornos como Visual RPG para .NET y se ha proporcionado la capacidad de conexión con base de datos estructuradas como el caso de DB2 de IBM con el RPG/400.

De hecho la comunidad de programadores de IBM i, donde RPG sigue siendo uno de los lenguajes de programación más usado, propuso a IBM renombrar el lenguaje de programación en el momento de publicar la nueva sintaxis donde se elimina la restricción de la posición fija e IBM declinó la propuesta (Woodie, 2020).

En definitiva, la comunidad sigue aún activa y hay mucho material para aprender a programar tanto para versiones como RPG II como RPG/400 y si tienes la necesidad o la oportunidad de trabajar con IBM i, entonces de seguro te toparás con RPG.

Glosario

Este glosario incluye algunas palabras empleadas a lo largo del libro y cuya definición sea importante mantener localizada.

ACE

Automatic Computing Engine o Máquina de Computación Automática. Fue la Máquina desarrollada en Reino Unido por Alan Turing.

ACM

Association for Computing Machinery o Asociación de Maquinaria de Computación.

AED-0

Automated Engineering Design o Diseño de Ingeniería Automatizado, aunque las siglas también se entendieron como *ALGOL Extended for Design* o ALGOL Extendido para el Diseño.

AIMACO

Air Material Compiler o Compilador de Material Aéreo.

ALGOL

ALGOrithmic Language o Lenguaje Algorítmico fue un lenguaje creado por un conjunto de científicos dedicados a la computación de EE.UU. y Europa. Ver el Capítulo 3.

ANSI

American National Standards Institute o Instituto de Estándares Nacional Americano. Una institución de estándares estadounidense.

ASCII

American Standard Code for Information Interchange o Código Estándar Americano para el Intercambio de Información fue desarrollado por los Laboratorios Bell en 1968.

BASIC

Beginners All-Purpose Symbolic Instruction Code o Código de Instrucciones Simbólico para Todo-Propósito para Principiantes.

BBN

Bolt, Beranek y Newman, empresa creada por los Richar Bolt, Leo Beranek y Robert Newman.

BCD

Binary-Coded Decimal o Decimal Codificado en Binario. Desarrollado y usado principalmente por IBM en código binario de 6 bits para codificar números, letras y caracteres especiales.

BCPL

Basic CPL o CPL Básico. Fue una simplificación de [CPL].

BEMA

Business Equipment Manufacturers Association o Asociación de Fabricantes de Equipos de Negocio.

BNF

Backus Normal Form o *Backus-Naur Form*, Forma Normal de Backus o Forma de Backus-Naur. Ver la Sección 3.2 en el Capítulo 3.

CAI

Computer-Aided Instruction o Instrucción Asistida por Computador.

CAL

Conversational Algebraic Language o Lenguaje Algebraico Conversacional.

CISL

Cambridge Information Systems Laboratory o Laboratorio de Sistemas de Información de Cambridge.

COBOL

Common Business Oriented Language o Lenguaje Común Orientado a los Negocios.

CODASYL

Conference On Data Systems Languages o Conferencia sobre Lenguajes de Sistemas de Datos.

COMTRAN

Comercial Translator o Traductor Comercial. Un lenguaje desarrollado por IBM para ser amistoso al mundo de los negocios.

CORAL

Computer On-line Real-time Applications Language o Lenguaje de Aplicaciones de Computador en Línea y en Tiempo-Real y muy posiblemente en sus primeras versiones *Computer On-line Radar Applications Language* o Lenguaje de Aplicaciones de Computador en Línea para Radar.

COWSEL

COntrolled Working SpacE Language o Lenguaje del Espacio de Trabajo Controlado.

CPL

Combined Programming Language o Lenguaje de Programación Combinado. En verdad el nombre sugiere muchas más posibles interpretaciones (ver Capítulo 12).

CTSS

Compatible Time-Sharing System o Sistema de Tiempo-Compartido Compatible. También se conoce como *Caltech Time-Sharing System* o Sistema de Tiempo-Compartido de Caltech por el Instituto de Tecnología de California donde fue creado.

DEC

Digital Equipment Corporation o Corporación de Equipos Digitales presidida inicialmente por Ken Olsen. Una gran empresa de computación entre la década de 1960 y 1990. En 1998 se fusionó a Compaq perdiendo su nombre en 2002 en otra fusión con Hewlett-Packard (HP).

DOPE

Dartmouth Oversimplified Programming Experiment o Experimento de Programación Sobre-simplificado de Dartmouth.

DSL

Domain Specific Language o Lenguaje de Dominio Específico. Son lenguajes que se crean para un propósito específico y su sintaxis y su semántica se diseñan para facilitar la expresión del dominio al que pertenecen.

ECMA

European Computer Manufacturers Association o Asociación Europea de Fabricantes de Computadores.

EPL

Early PL/I o PL/I temprano. Fue un subconjunto del lenguaje de programación PL/I desarrollado por McIlroy usando TMG para disponer del lenguaje PL/I en los sistemas operativos CTSS y GECOS empleados para crear Multics.

FARGO

Fourteen-o-one Automatic Report Generation Operation u Operación de generación automática de informes 1401. Es considerada la versión [RPG] 0.

FLPL

FORTRAN List Processing Language o Lenguaje de Procesamiento de Listas FORTRAN.

FOCAL

Formulating On-line Calculations in Algebraic Language o Formulación de Cálculos en línea en Lenguaje Algebraico. Aunque sus siglas también pudieron ser para *FOrmula CALculator* o Calculadora de Fórmulas.

FORMAC

FORmula MAnipulation Compiler o Compilador de Manipulación de Fórmulas.

GAMM

Gesellschaft für Angewandte Mathematik und Mechanik o Asociación de Matemática Aplicada y Mecánica.

GAT

Generalized Algebraic Translator o Traductor Algebraico Generalizado.

GE

General Electric.

GPM

General Purpose Macro-generator o Macro-generador de Propósito General. Ver la Sección 2.4.1 en el Capítulo 2.

GUIDE

Guidance for Users of Integrated Data-Processing Equipment u Orientación para Usuarios de Equipos Integrados de Procesamiento de Datos. Un grupo de usuarios de voluntarios de computadores y sistemas de IBM.

IA

Inteligencia Artificial.

IAL

International Algebraic Language o Lenguaje Algebraico Internacional. Fue el primer nombre otorgado al lenguaje que sería después nombrado como [ALGOL].

IBM

International Business Machines o Máquinas de Negocios Internacionales. Una empresa surgida a principios del siglo XX por la evolución de las empresas dedicadas a las máquinas de tabulación.

IFIP

International Federation for Information Processing o Federación Internacional para el Procesamiento de Información. Fue la federación a cargo de la definición de ALGOL 60 y ALGOL 68.

IPL

Information Processing Language o Lenguaje de Procesamiento de Información.

ISWIM

If you See What I Mean, Si ves lo que quiero decir. En el Capítulo 18.

IT

Internal Translator o Traductor Interno.

IVSYS

IVerson SYStem o Sistema de Iverson.

JOHNNIAC

John von Neumann Numerical Integrator and Automatic Computer o Integrador Numérico y Computadora Automática John von Neumann.

JOSS

JOHNNIAC Open Shop System o Sistema de Tienda Abierta JOHNNIAC fue un servicio de tiempo-compartido experimental de Rand Corporation diseñado para novatos en el mundo de la computación.

MAC

Proyecto MAC por sus siglas *Mathematics and Computation* o Matemáticas y Computación. Un proyecto conducido en la Universidad de Massachusetts en 1966 y del que se originaron algunas piezas clave como MacLisp.

MAD

Michigan Algorithm Decoder o Decodificador Algorítmico de Michigan.

MGH

Massachusetts General Hospital o Hospital General de Massachusetts.

MIT

Massachusetts Institute Technology o Instituto de Tecnología de Massachusetts.

MPPL

MultiPurpose Programming Language o Lenguaje de Programación MultiPropósito.

MUMPS

Massachusetts General Hospital Utility Multi-Programming System o Sistema de Programación-Múltiple de Servicios Públicos del Hospital General de Massachusetts.

NCSS

National Computer Software Systems o Sistemas de Software Computacional Nacionales.

NDRE

Norwegian Defense Research Establishment o Establecimiento Noruego de Investigación de Defensa.

NIH

National Institute of Health o Instituto Nacional de Salud.

NPL

National Physics Laboratory o Laboratorio de Física Nacional. En Reino Unido.

NRAO

National Radio Astronomy Observatory o Observatorio Nacional de Radioastronomía.

NSF

National Science Foundation o Fundación Nacional de Ciencias.

PCM

Pulse-Code Modulation o Modulación por Impulsos Codificados.

PILOT

Programmed Inquiry, Learning or Teaching o Investigación, aprendizaje o enseñanza programada.

PLATO

Programmed Logic for Automatic Teaching Operations o Lógica Programada para Operaciones de Enseñanza Automática.

POP

Package for Online Programming o Paquete para Programación en Línea. El nuevo nombre para el lenguaje [COWSEL].

REFAL

Recursive Functions Algorithmic Language o Lenguaje Algorítmico de Funciones Recursivas.

REPL

Read, Eval, Print and Loop o Lee, Evalúa, Imprime y Bucle. Indica la interacción de la máquina con el usuario, lee la instrucción, la evalúa, imprime el resultado y vuelta a comenzar.

RPG

Report Program Generator o Generador de Programas de Informes.

Ver el Capítulo 1.

SAP

Symbolic Assembly Program o Programa Ensamblador Simbólico. Ver el Capítulo 2.

SET

SET Language o Lenguaje de Conjuntos.

SCALP

Self Contained ALgol Processor o Procesador ALGOL Autocontenido.

SHARE

Grupo de usuarios voluntarios para computadores mainframe de IBM fundado en 1955.

SIL

SNOBOL Implementation Language o Lenguaje de Implementación de SNOBOL.

SLAC

Stanford Linear Accelerator o Acelerador Lineal de Stanford.

SQL

Structured Query Language o Lenguaje de Consultas Estructurado. Un lenguaje que podemos considerar de cuarta generación, declarativo o incluso de dominio específico aunque en nuestros días haya evolucionado hasta ser Turing completo.

STC

Standard Telephones and Cables o Cables y Teléfonos Estándares. Fue una empresa de telecomunicaciones de Reino Unido pionera en tecnologías como la fibra óptica.

THE

Technische Hogeschool Eindhoven o Universidad de Tecnología de Eindhoven. Se empleó como el nombre del sistema operativo

liderado por Edsger W. Dijkstra.

TMG

TransMoGrifiers un lenguaje creado por McClure en 1965 para escribir compiladores.

UMMPS

University of Michigan Multi-Programming Supervisor o Supervisor de Multi-Programación de la Universidad de Michigan.

URSS

Unión de Repúblicas Socialistas Soviéticas también conocida como Unión Soviética (US) era un conjunto de hasta 15 Repúblicas Socialistas Soviéticas (RSS) formadas desde 1922 hasta su disolución en 1991 que tuvieron un papel importante en la Segunda Guerra Mundial contra la Alemania Nazi y en la Guerra Fría contra Estados Unidos (EEUU).

VDM

Vienna Development Method o Método de Desarrollo de Viena.

Bibliografía

Mucha de la información obtenida para la realización de este libro ha partido de una búsqueda en Internet y principalmente la lectura de entradas del proyecto Wikipedia. También se han adquirido algunos documentos, tesis y publicaciones de sitios como ACM Digital Library para contrastar información contradictoria recurriendo directamente a las fuentes.

Muchos de esos artículos revisados, sitios web o libros consultados se encuentran a continuación.

Abrahams, P. W., Barnett, J. A., Book, E., Firth, D., Kameny, S. L., Weissman, C., Hawkinson, L., Levin, M. I., & Saunders, R. A. (1966). The LISP 2 programming language and system. *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, 661-676. doi.org/10.1145/1464291.1464362

Abrams, P. S. (1966). *An interpreter for "Iverson notation"*. Stanford University.

Abrams, P. S. (1975). What's wrong with APL? *Proceedings of Seventh International Conference on APL*, 1-8. doi.org/10.1145/800117.803777

Arden, B. W., Galler, B. A., & Graham, R. M. (1961). MAD at Michigan: its function & features. *Datamation, 7*(12), 27-28.

Arden, B., & Graham, R. (1959). On GAT and the Construction of Translators. *Commun. ACM, 2*(7), 24-26. doi.org/10.1145/368370.368373

Bachelor, G. A., Dempster, J. R. H., Knuth, D. E., & Speroni, J. (1961). SMALGOL-61. *Commun. ACM*, *4*(11), 499-502. doi.org/10.1145/366813.366843

Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., Woodger, M., & Naur, P. (1960). Report on the Algorithmic Language ALGOL 60. *Commun. ACM*, *3*(5), 299-314. doi.org/10.1145/367236.367262

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference. *IFIP Congress*. api.semanticscholar.org/CorpusID:44764020

Barron, D. W., Buxton, J. N., Hartley, D. F., Nixon, E., & Strachey, C. (1963). The Main Features of CPL. *The Computer Journal*, *6*(2), 134-143. doi.org/10.1093/comjnl/6.2.134

Beechhold, H. F. (1993). *Vanilla PILOT*. *5*(37), 67-69.

Berner, R. W. (1957). How to consider a computer. *Automatic Control Magazine*, 66-69.

Berner, R. W. (1971). A View of the History of COBOL. *Honeywell Computer Journal*, *5*.

Beyer, K. W. (2012). *Grace Hopper and the Invetion of the Information Age*. Smithsonian Institution.

Boytchev, P. (2010). *Logo Tree Project*. www.edtechpolicy.org/cyberk12ARCHIVE/Resources/Microworlds/LogoTreeProject.pdf

Brock, D. C. (2020). *Discovering Dennis Ritchie's Lost Dissertation*. computerhistory.org/blog/discovering-dennis-ritchies-lost-dissertation/

Brodie, L. (1981). *Starting FORTH*. www.forth.com/wp-content/uploads/2018/01/Starting-FORTH.pdf

Caine, S. H., & Gordon, E. K. (1968). *TTM: An Experimental Interpretive Language*. California Institute of Technology.

Chatley, R., Donaldson, A., & Mycroft, A. (2022). The Next 7000 Programming Languages. En *Computing and Software Science* (pp. 250-282). Springer-Verlag. doi.org/10.1007/978-3-319-91908-9_15

Christopher, T. W. (1996). *EULER: An Experiment in Language Definition*. Illinois Institute of Technology.

Cocke, J., & Schwartz, J. T. (1970). *Programming Languages and Their Compilers*. Courant Institute of Mathematical Sciences.

Corbató, F. J., Merwin-Daggett, M., & Daley, R. C. (1962). An Experimental Time-Sharing System. *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*, 335-344. doi.org/10.1145/1460833.1460871

Corporation, C. D. (1977). *RPG II Version 2 Reference Manual*. doi.org/96768710

Corporation, I. B. M. (1964). *FARGO for IBM 1401* [Form]. Systems Reference Library. bitsavers.org/pdf/ibm/1401/C24-1464-3_1401_fargo.pdf

Davis, M., & Schonberg, E. (2011). *Jacob Theodore Schwartz*. www.settheory.com/Schwartz_Jacob_Martin_and_Ed.pdf

Duncan, F. G. (1967). ALGOL Bulletin No. 26. *SIGPLAN Not.*, *2*(11), 1-49. doi.org/10.1145/1139498.1139500

Evans, A. (1968). PAL—a language designed for teaching programming linguistics. *Proceedings of the 1968 23rd ACM National Conference*, 395-403. doi.org/10.1145/800186.810604

Farber, D. J. (1964). *635 Assembly System - GAP*. Bell Telephone Laboratories Computation Center.

Farber, D. J. (1971). A Survey of the Systematic Use of Macros in

Systems Building. *SIGPLAN Not.*, *6*(9), 29-36. doi.org/10.1145/942596.807057

Fonsecai, P., & Casanovas, C. J. (2009). JGPSS, An open source GPSS framework to teach simulation. *Proceedings of the 2009 Winter Simulation Conference (WSC)*, 256-267. doi.org/10.1109/WSC.2009.5429335

Forest, B. (1961). BALGOL at Stanford: a fast compiler on a slow computer. *Datamation*, *7*(12), 24-26.

Gough, J. (1993). *Watching the Skies: History of Groud Radar for the Defence of the United Kingdom by the Royal Air Force from 1946 to 1975*. Stationery Office Books.

Graham, R. M. (1958). Translation between Algebraic Coding Languages. *Preprints of Papers Presented at the 13th National Meeting of the Association for Computing Machinery*, 1-2. doi.org/10.1145/610937.610964

Harvey, B. (1997). *Computer Science Logo Style*. The MIT Press.

Hoare, C. A. R. (1961). Algorithm 63: Partition. *Commun. ACM*, *4*(7), 321. doi.org/10.1145/366622.366642

Hoare, C. A. R. (1962). Quicksort. *The Computer Journal*, *5*(1), 10-16. doi.org/10.1093/comjnl/5.1.10

Hoare, C. A. R. (1965). *A Programming Language for Processor Construction*.

Hoare, T. (2009). *Null References: The Billion Dollar Mistake*. www.infoq.com/presentations/Null-References-The-Billion-Dollar-Mistake-Tony-Hoare/

Holbrook, B. D., & Brown, W. S. (1982). *A History of Computing Research at Bell Laboratories (1937-1975)* [Computing Science Technical Report]. AT&T Bell Laboratories.

Holmevik, J. R. (1994). Compiling SIMULA: a historical study of technological genesis. *IEEE Annals of the History of Computing*, *16*(4), 25-37. doi.org/10.1109/85.329756

Höltgen, S., & Baranovska, M. (2022). *Hello, I'm Eliza: Fünfzig Jahre Gespräche mit Computern*.

Igarashi, S., Iwamura, T., Sakuma, K., Simauti, T., Simuzu, T., Takasu, S., Wada, E., & Yoneda, N. (1969). ALGOL N. *ALGOL Bull.*, *30*, 38-85.

Irons, E. T. (1970). Experience with an extensible language. *Commun. ACM*, *13*(1), 31-40. doi.org/10.1145/361953.361966

Iverson, K. E. (2000). A personal view of APL. *SIGAPL APL Quote Quad*, *30*(3), 4-13. doi.org/10.1145/360487.360477

Iverson, K. E. (1962). *A programming language*. John Wiley & Sons, Inc.

Kelly, B. (2009). *IBM RPG: A Great Language with a Greater History*. www.nicklitten.com/a-brief-history-of-the-ibm-rpg-programming-language/

Kurtz, T. E. (1978). BASIC Session. En *History of Programming Languages* (pp. 515-550). Association for Computing Machinery. doi.org/10.1145/800025.1198360

Lampson, B. (2011). *A Culture of Innovation* (D. Walden & R. Nickerson, eds.). Waterside Publishing.

Landin, P. J. (1966). The next 700 programming languages. *Commun. ACM*, *9*(3), 157-166. doi.org/10.1145/365230.365257

Ltd, F. (1968). *FM1600B Microcircuit Computer*. www.sba.unipi.it/sites/default/files/2015_05_29_08_44_13.pdf

McCarthy, J. (1959). *Memorandum to P. M. Morse Proposing Time-Sharing*. jmc.stanford.edu/computing-science/timesharing-memo.html

McCarthy, J. (1960). Recursive functions of symbolic expressions and

their computation by machine, Part I. *Commun. ACM, 3*(4), 184-195. doi.org/10.1145/367177.367199

McCarthy, J. (1962). Time Sharing Computer Systems. *Management and the Computer of the Future*, Chapter 6.

McCarthy, J. (1978). History of LISP. En *History of Programming Languages* (pp. 173-185). Association for Computing Machinery. doi.org/10.1145/800025.1198360

McIlroy, M. D. (1960). Macro Instruction Extensions of Compiler Languages. *Commun. ACM, 3*(4), 214-220. doi.org/10.1145/367177.367223

Mills, D. L. (1968). *The Syntactic Structure of MAD/I*. University Michigan. apps.dtic.mil/sti/citations/AD0671683

Mooers, C. N., Deutsch, L. P., & Floyd, R. W. (1965). Programming Languages for Non-Numeric Processing—1: TRAC, a Text Handling Language. *Proceedings of the 1965 20th National Conference*, 229-246. doi.org/10.1145/800197.806048

Moore, C. H. (1970). *Programming a Problem-oriented Language*. colorforth.github.io/POL.html

Nelson, T. (1974). *Computer Lib/Dream Machines*.

of California, O. A. (1998). *Register of the John A. Starkweather Papers, 1965-1985*. oac.cdlib.org/view?docId=tf2d5nb1xg;style=oac4;doc.view=entire_text

Papert, S. (1980). *Mindstorms*. Harvester Press.

Perlis, A. J. (2008). *ALGOL, More than just ALGOL*. api.semanticscholar.org/CorpusID:47672972

Popplestone, R. (1999). *The Early Development of POP*. www-robotics.cs.umass.edu/Popplestone/pop_development.html

Pouzin, L. (1965). *The SHELL: A Global Tool for Calling and Chaining Procedures in the System*. Massachusetts Institute of Technology. people.csail.mit.edu/saltzer/Multics/Multics-Documents/MDN/MDN-4.pdf

Pouzin, L. (2000). *The Origin of the Shell*. multicians.org/shell.html

Radin, G. (1978). The early history and characteristics of PL/I. *SIGPLAN Not.*, *13*(8), 227-241. doi.org/10.1145/960118.808389

Rather, E. D., Colburn, D. R., & Moore, C. H. (1993). The evolution of Forth. *The Second ACM SIGPLAN Conference on History of Programming Languages*, 177-199. doi.org/10.1145/154766.155369

Rawlings, N. (2014). The History of NOMAD: A Fourth Generation Language. *IEEE Annals of the History of Computing*, *36*(1), 30-38. doi.org/10.1109/MAHC.2014.10

Richards, M. (2012). How BCPL Evolved from CPL. *The Computer Journal*, *56*(5), 664-670. doi.org/10.1093/comjnl/bxs026

Ritchie, D. M. (1968). *Program Structure and Computacional Complexity*. archive.computerhistory.org/resources/access/text/2020/05/102790971/Ritchie_dissertation.pdf

Ritchie, D. M. (1993). The development of the C language. *The Second ACM SIGPLAN Conference on History of Programming Languages*, 201-208. doi.org/10.1145/154766.155580

Rubio Jiménez, M. Á. (2021). *Historia de los Lenguajes de Programación: años 1940-1959*. Altenwald Books.

Sammet, J. E. (1969). *Programming Languages: History and Fundamentals*. Prentice-Hall, Inc. doi.org/10.5555/1096897

Sammet, J. E. (1978). The early history of COBOL. En *History of Programming Languages* (pp. 199-243). Association for Computing Machinery. doi.org/10.1145/800025.1198367

Serrão, R. G. (2022). *Why APL is a language worth knowing*. mathspp.com/blog/why-apl-is-a-language-worth-knowing

Shapiro, J. S. (2004). Extracting the Lessons of Multics. *login Usenix Mag.*, *29*(6). www.usenix.org/publications/login/december-2004-volume-29-number-6/extracting-lessons-multics

Sherwood, B. (1974). *The TUTOR Language*.

Starkweather, J. A. (1967). *Computer-Assisted Learning in Medical Education*. *97*.

Steele, G. L., & Gabriel, R. P. (1996). The evolution of Lisp. En *History of Programming Languages—II* (pp. 233-330). Association for Computing Machinery. doi.org/10.1145/234286.1057818

Strachey, C. (1965). A general purpose macrogenerator. *The Computer Journal*, *8*(3), 225-241. doi.org/10.1093/comjnl/8.3.225

Strachey, C. S. (1959). Time sharing in large, fast computers. *IFIP Congress*. api.semanticscholar.org/CorpusID:5144680

Thompson, K. (1972). *Users' Reference to B*. Bell Telephone Laboratories. www.bell-labs.com/usr/dmr/www/kbman.pdf

Turchin, V. F. (1979). A supercompiler system based on the language REFAL. *SIGPLAN Not.*, *14*(2), 46-54. doi.org/10.1145/954063.954069

van Wijngaarden, A. (1963). *Generalized Algol* (Número MR 57/63/R).

Ware, ed., Willis H. (1960). *Soviet Computer Technology - 1959*. RAND Corporation.

Wayne, H. (2020). *10 Most(ly Dead) Influential Programming Languages*. www.hillelwayne.com/post/influential-dead-languages/

Wirth, N., & Weber, H. (1966). EULER: A generalization of ALGOL and its formal definition: Part 1. *Commun. ACM*, *9*(1), 13-25. doi.org/10.1145/365153.365162

Wirth, N., & Weber, H. (1966). EULER: a generalization of ALGOL, and its formal definition: Part II. *Commun. ACM, 9*(2), 89-99. doi.org/10.1145/365170.365202

Woodie, A. (2020). *Is it time to rename RPG?*. www.itjungle.com/2020/08/24/is-it-time-to-rename-rpg/

Woodward, P. M., Wetherall, P. R., & Gorman, B. (1973). *Official Definition of CORAL 66*. archive.org/details/official-definition-of-coral-66

Yngve, V. H. (1957). A Framework for Syntactic Translation. *Mechanical Translation, 4*(3), 59-65. aclanthology.org/www.mt-archive.info/50/MT-1957-Yngve.pdf

Yngve, V. H. (1962). COMIT as an IR language. *Commun. ACM, 5*(1), 19-28. doi.org/10.1145/366243.366720

Yost, J. R. (2014). *An Interview with Butler Lampson*. Charles Babbage Institute.

Cohen, I. B., Welch, G. W., & Campbell, R. V. D. (eds.). (1999). *Makin' numbers: Howard Aiken and the computer*. MIT Press.

Early Multics Development and the MSPM. (1965-1969). multicians.org/mspmtoc.html

PL/I: Language Specifications. (1965). IBM. bitsavers.org/pdf/ibm/360/pli/C28-6571-1_PL_I_Language_Specifications_Jul65.pdf