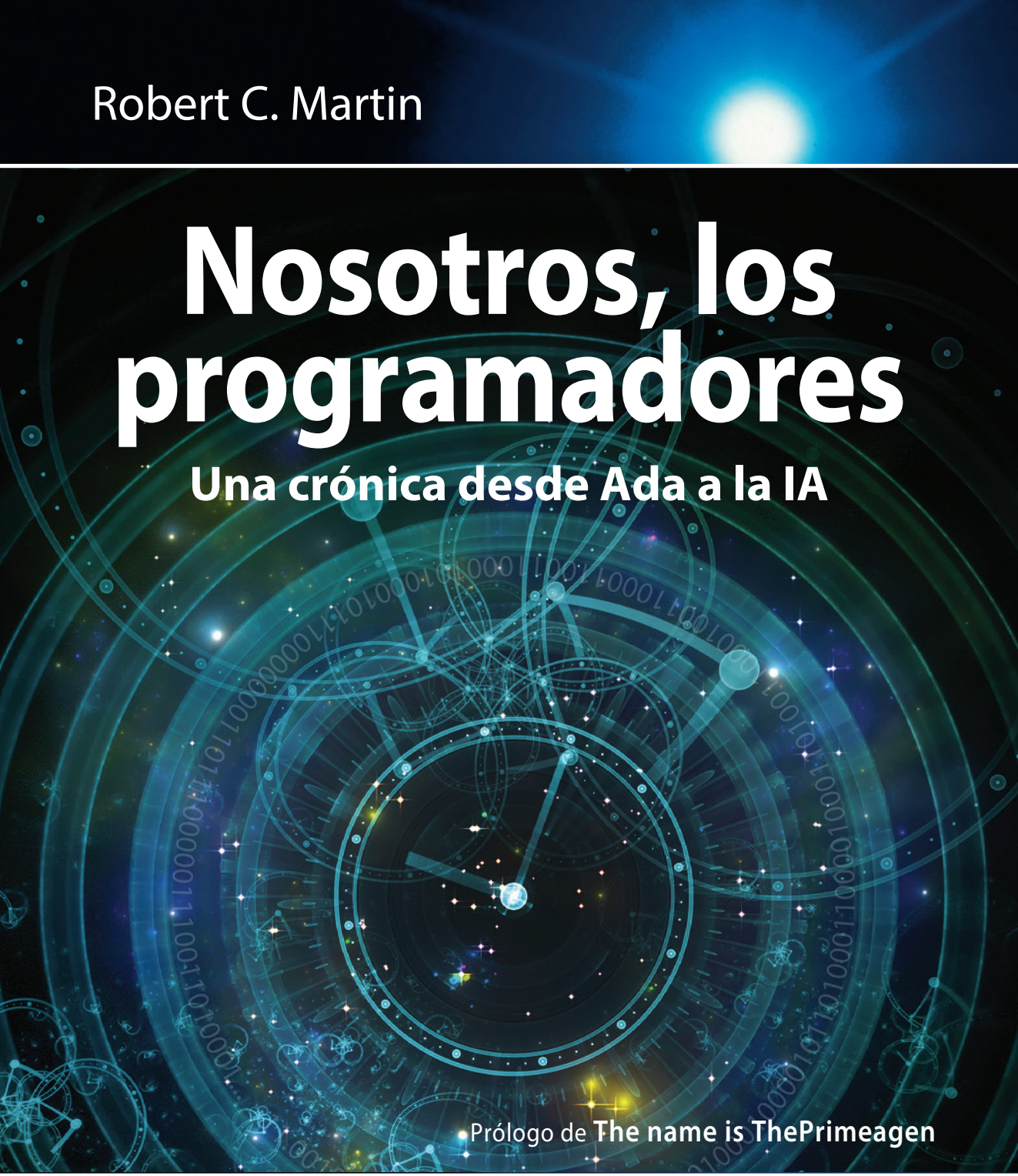
Robert C. Martin

Nosotros, los programadores

Una crónica desde Ada a la IA



Prólogo de **The name is ThePrimeagen**

ANAYA MULTIMEDIA

Índice de contenidos

I

¿Quiénes somos?

Nosotros, los programadores, somos los que hablamos a las máquinas y las hacemos funcionar. Somos los que hacemos que cobren vida, y así damos vida a nuestras economías y sociedades. No pasa nada en el mundo sin nosotros. ¡Nosotros gobernamos el mundo!

Otras personas creen que gobiernan el mundo y que después nos pasan esas reglas a nosotros y nosotros escribimos las reglas que se ejecutan en las máquinas que lo gobiernan todo. Pero esta posición ascendente y necesaria no siempre ha sido así. En los primeros días de la programación, los programadores eran invisibles. Todas las miradas estaban puestas en los ordenadores y sus grandes promesas. Eran las máquinas, y aquellos que las construían, los que eran ascendentes e impresionantes. Nadie se fijaba en los programadores que simplemente hacían que las máquinas funcionasen. Éramos poco más que ruido de fondo.

Acerca de aquellos primeros años, Dijkstra dijo:[1]

> "Puesto que [cada ordenador] era una máquina única, [el programador] sabía demasiado bien que sus programas solo tenían un significado local y también, puesto que era muy evidente que esta máquina tendría una vida limitada, sabía que muy poco de su trabajo tendría un valor duradero".

1. *The Humble Programmer*, ACM Turing Lecture, 1972.

Al final, sin embargo, por recomendación de George Biddell Airy, el astrónomo real, el gobierno rehusó dar más financiación. La carta de la oficina del primer ministro decía:[12]

> "Los proyectos del señor Babbage parecen ser tan indefinidamente caros, su éxito final tan problemático y su gasto tan grande y tan imposible de calcular, que el gobierno no tendría justificación para asumir más responsabilidad".

Así, después de los gastos de una década, y de 17.000 libras esterlinas, se abandonó el esfuerzo.

En mi humilde opinión, aunque creo que es probable que Babbage lo negase, creo que el fracaso definitivo se debió a que su atención se dispersaba continuamente hacia otras ideas más ambiciosas. Para Babbage, acabar no era tan divertido como empezar.

El argumento de la tecnología

Se ha dicho que el fracaso a la hora de completar la Máquina Diferencial se debió a la incapacidad de la tecnología de metalistería de principios del siglo XIX para crear piezas con la precisión necesaria. Sin embargo, no hay pruebas reales que respalden esta versión. Las piezas supervivientes son muy precisas y hay prototipos de la máquina que siguen funcionando hoy en día.

Si Babbage hubiese tenido la voluntad, la concentración y los recursos para terminar esa máquina, hay motivos para pensar que habría funcionado tal y como estaba diseñada. Aunque, como veremos, conseguir que la máquina funcione de verdad es muy diferente a construirla sin más.

La Máquina Analítica

> *... y, sin embargo, en la inmensidad de mis circuitos rebosantes puedo recorrer los infinitos cauces de las probabilidades futuras y reconocer que un día existirá un ordenador cuyos meros parámetros operativos no seré digno de calcular, aunque será mi destino llegar a diseñarlo.*
>
> —Pensamiento Profundo, *Guía del autoestopista galáctico*

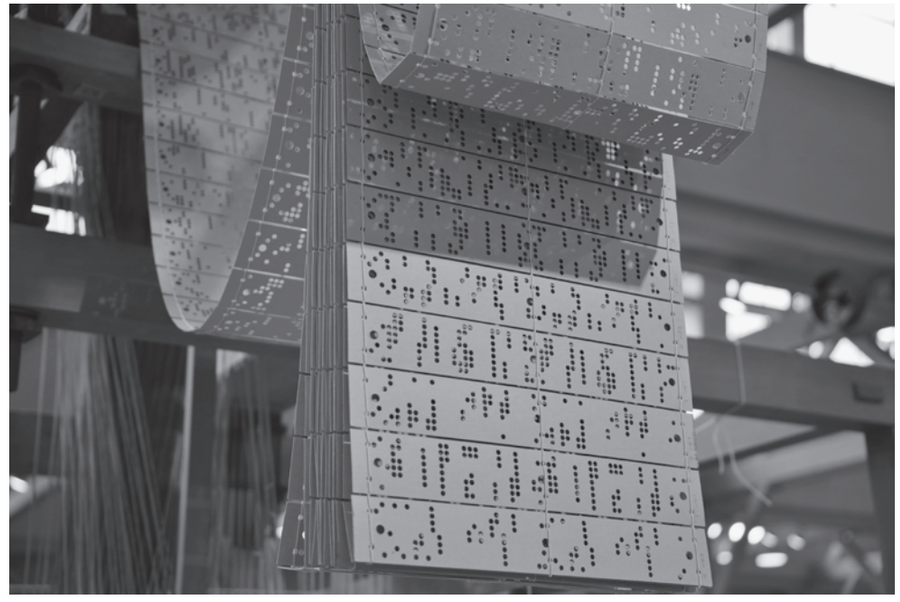
12. Swade, p. 176.

¿Qué era lo que estaba distrayendo a Babbage de la compleción de la Máquina Diferencial?

Imagine una máquina del tamaño de una locomotora. En vez de 6 columnas de 20 dígitos, tiene 1.000 columnas de 50 dígitos. Tiene un *bus* mecánico que transfiere los valores de esas columnas a los dos canales de salida de lo que Babbage llamó "el molino" (*mill*). El molino puede sumar, restar, multiplicar y dividir esos números en precisión simple o doble (de 100 dígitos). Entonces, el *bus* transfiere los valores resultantes de vuelta al almacén de 1.000 columnas.

Imagine que esta máquina está controlada por un conjunto de instrucciones en tarjetas perforadas que están unidas formando una cadena, como las tarjetas perforadas en un telar de Jacquard. Esas tarjetas de instrucciones dirigen el *bus* para extraer valores de columnas particulares del almacén y moverlas por el *bus* a las salidas del molino. Ordenan al molino que opere sobre los valores proporcionados por el *bus*. Ordenan al *bus* que mueva los resultados desde el modelo de vuelta al almacén.

Cargan constantes en el almacén. Ordenan que los valores se impriman en una impresora, se representen en un gráfico a través de un trazador o hagan sonar una simple campana.



El hombre que ideó la arquitectura de ordenadores que combinaba instrucciones y datos fue Alan Turing, pero fue la influencia de John von Neumann la que guio la adopción de esa arquitectura.

La historia de estos dos hombres y la sinergia de sus ideas es un relato digno de hogueras de campamento y nubes asadas. Y en el humo sobre la hoguera da vueltas el fantasma de David Hilbert.

DAVID HILBERT

El origen del auge de la informática en el siglo XX puede encontrarse en el fracaso abyecto de un hombre concreto: David Hilbert.

De todos los matemáticos de principios del siglo XX, es posible que no hubiese ninguno mejor considerado que Hilbert. Desde 1895 hasta el ascenso del partido nazi, la universidad de Gotinga, donde Hilbert era profesor de matemáticas, era el centro del mundo matemático, su círculo de colaboradores y estudiantes incluía a celebridades como Felix Klein, Hermann Weyl, Emanuel Lasker, Alonzo Church, Emmy Noether, Hermann Minkowski y John von Neumann.

Hilbert adoptó y defendió la teoría de conjuntos de Georg Cantor y los números transfinitos. En aquella época, no era una postura muy popular y Hilbert tuvo que aguantar un buen chaparrón. Pero, al final, fue la visión que prevaleció.

Estoy seguro de que la mayoría de los lectores recordarán los rudimentos de la teoría de conjuntos. Causó furor entre los profesores de matemáticas de primaria en los sesenta. Formaba parte de las "nuevas matemáticas". Es probable que sean menos los que recuerden qué son los números transfinitos, si es que llegaron a aprenderlos.

Cantor demostró que hay más de un tipo de infinito. De hecho, demostró que hay un número infinito de infinitos, cada uno de ellos un poco más grande que el siguiente. Los dos infinitos a los que estamos más acostumbrados son el infinito de los números contables y el infinito del continuo.

Es posible colocar todos los números racionales y algebraicos en una correspondencia uno a uno con los números naturales, demostrando así que el conjunto de todos los números es contablemente infinito. Cantor demostró que ninguna correspondencia uno a uno así es posible con el conjunto de todos los números reales, probando por tanto que el tamaño del conjunto de todos los números reales es mayor que el del conjunto de todos los números naturales.

NOTA: Como comentario adicional, me parece gracioso que nuestras dos teorías principales sobre la realidad, la mecánica cuántica y la relatividad general, estén alineadas con uno de estos dos infinitos, y que sea ese alineamiento el que hace que las dos teorías sean incompatibles de una forma tan frustrante. Como verá más adelante, fue Von Neumann quien cerró la brecha entre estos dos infinitos en el caso muy específico de la falta de correspondencia entre la ecuación de Schrödinger y el análisis matricial de Heisenberg.

Hilbert estaba fascinado por la idea de axiomatizar[1] las matemáticas del mismo modo que Euclides había axiomatizado la geometría del plano. En 1899, Hilbert publicó *Fundamentos de la geometría*, que axiomatizaba geometrías no euclidianas con una formalidad que superaba en mucho a la de Euclides y que estableció el estándar para los formalismos matemáticos a partir de entonces.

Pero Hilbert no estaba satisfecho con solo axiomatizar la geometría. Quería aplicar el mismo nivel de formalismo a todas las matemáticas, derivándolo a partir de unos pocos axiomas fundamentales. Argumentaba que toda pregunta matemática tenía una respuesta definida que podía deducirse a partir de esos axiomas.

Las palabras grabadas en su lápida son: *Wir müssen wissen. Wir werden wissen* ("Debemos saber. Sabremos").

Pero, después del éxito de Hilbert con la geometría, empezaron a aparecer las primeras grietas en su objetivo para las matemáticas. En 1901 Bertrand Russell demostró que era posible, al utilizar el formalismo de la teoría de conjuntos, expresar un enunciado que no era ni verdadero ni falso. Contradiciendo la afirmación de Hilbert de que "sabremos", Russell creó un enunciado[2] que era incognoscible.

Hilbert alentó desesperadamente a los matemáticos de todo el mundo a rescatar la teoría de conjuntos de la catástrofe de Russell, exclamando: "Nadie nos echará de este paraíso que Cantor ha creado para nosotros". La cuestión se volvió tan

1. Axiomatizar es crear un conjunto de axiomas o postulados y un conjunto de reglas lógicas a partir de los cuales puede derivarse por completo el área que se estudia.
2. Ese enunciado puede parafrasearse como: "El conjunto de todos los conjuntos que no se contienen a sí mismos ¿se contiene a sí mismo?". O, por decirlo de manera más simple: este enunciado es falso.

El Laboratorio de Investigación Balística (*Ballistics Research Lab*, BRL) se creó para abordar estas cuestiones. Al principio, utilizaban el mismo enfoque con el que estaba familiarizado Babbage; habitaciones llenas de "computadores", sobre todo mujeres, armados con calculadoras de sobremesa, haciendo sumas y multiplicaciones sin parar.

Viendo los problemas a los que se enfrentaban, y pensando en el futuro, Von Neumann predijo: "Va a haber un avance en las máquinas de cálculo que tendría que funcionar en parte como el cerebro. Esas máquinas se conectarían a todos los sistemas grandes, como los sistemas de telecomunicaciones, tendidos eléctricos y grandes fábricas".[21] Era un sueño, un germen de una idea. Emocionante, pero incompleto: era demasiado para que ese germen empezase a creer.

En septiembre de 1940, Von Neumann fue nombrado miembro de la junta de asesores del BRL. Para diciembre, también se había convertido en el asesor jefe de balística del comité de preparación para la guerra de la American Mathematical Society. En resumen, estaba muy solicitado.

A lo largo de los dos años siguientes, Von Neumann se convirtió en un experto en las ondas de choque creadas por las municiones, incluyendo la carga hueca. A finales de 1942, fue enviado a Inglaterra en una "misión secreta". No se sabe mucho sobre esto, ni siquiera en la actualidad, pero está claro que estaba aprendiendo más acerca de las ondas de choque explosivas.

La máquina NCR

Durante este viaje, Von Neumann vio la máquina de contabilidad NCR en acción en la Oficina del Almanaque Náutico de Bath. Este dispositivo era una calculadora mecánica con un teclado, una impresora y seis registros. Era capaz de realizar ~200 sumas por hora. No era programable, pero, debido a los registros y un ingenioso mecanismo de tabulación, un operador podía recorrer una secuencia de operaciones repetitivas con relativa rapidez. Von Neumann estaba tan intrigado que en el tren de vuelta a Londres escribió un "programa" de aproximación mejorado para la máquina.

Unos meses antes de su precipitado regreso a Princeton, Von Neumann escribió que había "desarrollado un interés obsceno en las técnicas de computación". Si ese interés se vio estimulado por la máquina NCR o por un posible encuentro con Turing sigue siendo objeto de debate.

21. Bhattacharya, p. 103.

La evidencia es escasa, pero hay una probabilidad razonable de que Turing y Von Neumann se reuniesen durante ese periodo y hablasen de maquinaria de computación. Quizá fue la unión de ideas de estos dos hombres la que fertilizó el germen en la mente de Von Neumann para que empezase a moverse y crecer.

Los Álamos: el Proyecto Manhattan

En julio de 1943, mientras todavía estaba en Inglaterra, Von Neumann recibió una carta urgente que decía: "Necesitamos su ayuda desesperadamente". La carta estaba firmada por J. Robert Oppenheimer.

Von Neumann ya había contribuido, sin saberlo, al Proyecto Manhattan debido a su investigación sobre las ondas de choque explosivas. Había demostrado que las explosiones en el aire eran más destructivas que en tierra y mostraba cómo calcular la altitud óptima.

Llegó a Los Álamos en septiembre y se puso manos a la obra. La necesidad desesperada que había mencionado Oppenheimer tenía que ver con el diseño teórico del arma de implosión de plutonio. La experiencia de Von Neumann con la carga hueca le llevó a sugerir rodear el núcleo de plutonio con cargas en forma de gajos que concentraría las ondas de choque hacia dentro de manera esférica.

El ejército y la Marina afirmaron que su trabajo sobre ondas de choque y balística fue tan importante como su trabajo con la bomba. Por tanto, Von Neumann tenía el privilegio exclusivo de ir y venir de Los Álamos cuando le apeteciese. Y eso le dio una visión única del entorno informático de EE. UU. que nadie más tenía.

El dispositivo de implosión debía modelarse y la carga de cálculo era enorme. Así pues, se compraron a IBM diez calculadoras con tarjetas perforadas.[22] Estos dispositivos se programaban con tableros de conexiones que especificaban los campos en las tarjetas, las operaciones que realizar en esos campos y dónde perforar los resultados.

Imagine un mazo de mil tarjetas, cada una con la posición inicial de una partícula dentro de la implosión. Imagine pasar ese mazo por una máquina para producir un mazo de mil tarjetas con los resultados y, después, pasar ese mazo por la siguiente máquina, y luego la siguiente, y después la siguiente. Y continuar esa operación 24 horas al día, seis días a la semana, semana tras semana.

22. "Imagínese un monstruo negro amenazador, que llena un cubo de casi dos metros cuando se cierra. La parte delantera era un reproductor 512 muy modificado: dos alimentadores de tarjetas de cien por minuto y dos apiladores. Había dos tableros de conexiones de doble panel cubiertos de cables, en la parte delantera, y un panel de interruptores numéricos en el lado derecho. Sujeta a la parte trasera de la perforadora había una caja de miles de relés de Lake y, acoplada a esa, una segunda caja". (Herb Grosch, `www.columbia.edu/cu/computinghistory/aberdeen.html`)

El Mark I funcionaba con un motor de 200 rpm, lo que le permitía sumar un registro a otro en 300 milisegundos. También tenía un motor aritmético aparte que podía multiplicar en 10 segundos, dividir en 16 segundos y resolver un logaritmo en 90 segundos. El aparato completo estaba controlado, como el telar de Jacquard, por instrucciones perforadas en una cinta de papel larga de algo más de 7 centímetros de ancho.

Aiken todavía no había oído hablar de la Máquina Analítica de Babbage, pero Babbage habría mirado esta máquina y se habría sentido íntimamente conectado a ella.

Esto era la Marina. El Mark I era un barco y Howard Aiken era el capitán. Y así era como dirigía esa operación. Todos iban a trabajar de uniforme y seguían el protocolo y la etiqueta militares. Había una guerra y todos eran soldados en ella.

Aiken no era un hombre fácil de tratar, y no todo el mundo apreciaba su estilo militar. Una de esas personas era Rex Seeber.[8] Seeber se unió al equipo en 1944, pero preguntó a Aiken si podría retrasar su incorporación para disfrutar de unas merecidas vacaciones. Aiken rechazó la solicitud y se sintió ofendido por ella.

A partir de entonces, sin importar lo duro que Seeber trabajase, Aiken ignoraba sus sugerencias e ideas. Cuando acabó la guerra, Seeber dimitió y aceptó un puesto en IBM para diseñar y construir la máquina que eclipsaría a Aiken y el Mark I.

Aiken estaba decepcionado porque la Marina hubiese puesto a Grace Hopper, una mujer oficial, como segunda al mando de su barco.[9] Cuando ella se presentó al servicio el 2 de julio de 1944, las primeras palabras que el comandante Aiken dirigió a la teniente júnior Grade Hopper fueron: "¿Dónde ha estado?".[10] Después, le ordenó que utilizase el Mark I para calcular los coeficientes de interpolación para calcular arcotangentes con una exactitud de 23 decimales. Le dio una semana.

En una semana, Hopper, una mujer que una vez dijo que no podía distinguir un ordenador de una cesta de tomates, tenía que programar una máquina de un tipo que pocos habían siquiera imaginado antes. Una máquina sin tutoriales de YouTube, ni siquiera un manual de instrucciones en papel. Lo único que el equipo de Aiken le dio fue un cuaderno improvisado con prisas que describía los códigos de instrucciones.

8. Robert Rex Seeber (1910-1969).
9. Beyer, p. 39.
10. Beyer, p. 39.

El Mark I se programaba perforando códigos en una cinta de papel. La cinta de papel constaba de filas horizontales con 24 posiciones. Cada posición podía tener un orificio en ella o no. Esto, por supuesto, se correspondía a 24 bits, aunque ellos no pensaban en eso de esta forma.

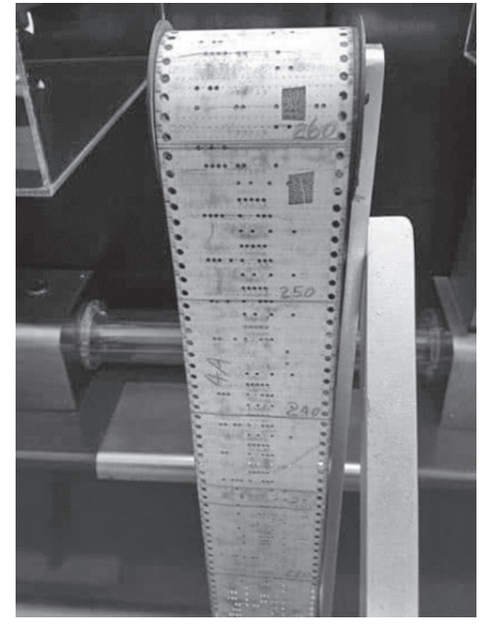
Los 24 bits se separaban en tres grupos de ocho. Los dos primeros grupos eran las direcciones binarias de los registros de memoria de entrada y salida. El tercero era la operación que había que realizar. La operación 0 era "sumar", así que `0x131400` era la instrucción para sumar el número en el registro `0x13` al número en el registro `0x14`.

Pero no era así como lo escribían. Lo escribían así: `| 521| 53| |`. Fíjese bien en esto y probablemente lo entenderá.

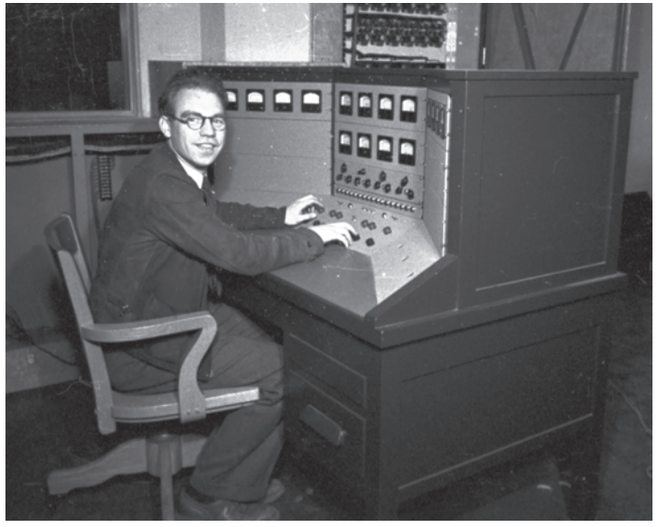
¿Ya lo tiene? ¿No? Es probable que esto le ayude. Los orificios en la cinta de papel serían:

```
| | | |o| | |o|o| | | |o| |o| | | | | | | | | |
 8 7 6 5 4 3 2 1 8 7 6 5 4 3 2 1 8 7 6 5 4 3 2 1
       1       3       1       4       0       0
```

Ahora lo ve, ¿no? ¿Ya está gritando?



Su padre era desagradable y distante. Puede que su madre, que murió antes de que John cumpliese 9 años, abusase de él sexualmente.[2] Su madrastra era una alcohólica neurótica que a veces gritaba a los transeúntes desde la ventana.



De adolescente, John era un abusón al que le gustaba maltratar a sus compañeros. Al final, lo enviaron a un internado,[3] donde se dedicaba a hacer el tonto y romper todas las reglas que podía. Suspendía una y otra vez, así que tenía que quedarse en la escuela de verano, donde pasaba el tiempo navegando y divirtiéndose. Rara vez volvía a casa.

Pese a sus malas notas, logró graduarse y se matriculó en la universidad de Virginia. Estudió química a instancias de su padre, pero no completaba el trabajo en el laboratorio y apenas asistía a sus clases, ya que prefería salir de fiesta. Al final, lo expulsaron.

Era 1943, así que lo llamaron a filas y, en el ejército, lo destinaron a Fort Stewart, en Georgia. Fue en una prueba de aptitud del Ejército donde comenzó el cambio. Clavó la prueba, así que el ejército lo envió a aprender ingeniería en la universidad de Pittsburgh.

2. Una sospecha inducida por el LSD que creció cuando tenía más de 60 años.
3. The Hill School en Pottstown, Pensilvania.

Superó sin dificultad los cursos previos de ingeniería, pasando mucho tiempo en los bares, pero no a costa de descuidar sus estudios, que había empezado a disfrutar.

Otra prueba de aptitud convenció al ejército de que tenía madera de médico, así que lo enviaron a estudiar medicina al Haverford College, donde tuvo un gran rendimiento.

Como parte de su formación como médico, hacía una residencia de 12 horas al día en un hospital de Atlantic City. Pero, después de unos meses, un bulto que tenía en la cabeza fue diagnosticado como un tumor de crecimiento lento, cuya extirpación le dejó un orificio en el cráneo cubierto por placas de metal que no encajaban bien. Eso lo liberó de sus obligaciones como residente y lo llevó a ser licenciado con honores del ejército en 1946.

Tras su licenciatura, para poner algo de distancia entre él y su familia, se matriculó en una escuela de medicina en la ciudad de Nueva York. Mientras estaba allí, ayudó a diseñar un recambio mejor ajustado para la placa de su cabeza. Pero la escuela de medicina le resultó decepcionante. Detestaba la memorización repetitiva por la fuerza y, al final, abandonó.

Así, se encontró de nuevo sin dirección, o casi. El único interés que aún tenía era crear un sistema Hi-Fi,[4] así que utilizó la GI Bill (una ley de reajuste de militares) para conseguir una plaza en una escuela de tecnología de radio. Allí, conoció al "primer profesor bueno" que había tenido.[5]

Con la ayuda de ese profesor, Backus se dio cuenta de que le gustaban de verdad las matemáticas y, además, se le daban bastante bien. Así pues, se matriculó en el programa de posgrado de matemáticas en la universidad de Columbia.

LUCES DE COLORES QUE HIPNOTIZAN

En la primavera de 1949, mientras estudiaba para su máster en Columbia, se topó con "una cosa interesante". Un amigo suyo le había dicho que echase un vistazo a la *Selective Sequence Electronic Calculator* (SSEC, calculadora electrónica secuencial selectiva) de IBM, situada en la sala de exposiciones de IBM en la Avenida Madison.

4. Un sistema de música de alta fidelidad, a menudo estéreo. Suele encontrarse en un tocadiscos y, quizá, en una radio.
5. Lorenzo, p. 24.

Tim Conrad y yo debatíamos sobre esos conceptos todo el tiempo. Escribíamos todo tipo de programas interesantes que tenían poco o nada que ver con el recorte por láser. En TAS, a menudo teníamos tiempo. Tiempo para pensar, para planear, para explorar, para... jugar.

Tim y yo elaboramos algoritmos de ordenación, algoritmos de búsqueda, esquemas de indexación y esquemas de cola. No teníamos *Algoritmos fundamentales* de Knuth (1968) porque no sabíamos que existía, pero inventamos o descubrimos juntos muchos de esos algoritmos mientras escribíamos código para este proyecto interesante o aquel o el de más allá. Fueron tiempos muy emocionantes.

Descubrí que el terminal de vídeo de Teradyne tenía una especie de direccionamiento de discurso primitivo. También era muy rápido, porque se conectaba directamente al *bus* de E/S del ordenador. Así pues, me puse a hacer lo que nadie había hecho en Teradyne hasta ese momento: diseñar formularios y diseños de pantalla y escribir código para ofrecer actualizaciones en tiempo real de esos formularios y pantallas. No eran exactamente interfaces gráficas de usuario, pero mi mente estaba llena de posibilidades alucinantes.

Mientras trabajaba en TAS vi mi primera calculadora de bolsillo: la HP-35. Me quedé asombrado con aquel dispositivo. Era pequeño. Era rápido. Calculaba raíces cuadradas y funciones trigonométricas. Tenía un lector LED. Una semana los ingenieros de nuestra oficina llevaban reglas de cálculo en el cinturón. A la semana siguiente, todos tenían HP-35. El cambio estaba en el aire y se acercaba a gran velocidad.

Pensándolo ahora, debería haberme quedado en TAS; era un entorno repleto de posibilidades. Pero las personas de 21 años no son precisamente sabias. El trayecto al trabajo era largo y molesto, y yo anhelaba un cambio de escenario. Así pues, acepté un empleo en Outboard Marine Corporation (OMC) en mi ciudad natal, Waukegan, Illinois.

1974

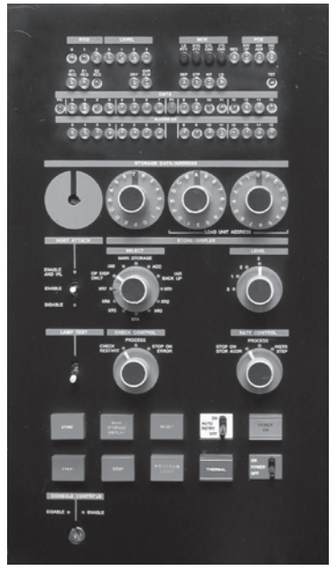
Acepté ese trabajo porque estaba cerca de casa, tanto que podía ir en bicicleta a la oficina. No voy a decir que el empleo no era un desafío a nivel técnico; sí lo era. Pero yo no encajaba bien en la cultura de esa empresa. Me di cuenta el primer día, cuando mi jefe me mandó llevar corbata a partir de ese momento.

OMC hacía cosas como cortadoras de césped Lawn Boy y motores fueraborda Johnson para barcos. Para fabricar los motores, construyeron unas instalaciones enormes para la fundición de aluminio a presión. Caminar por aquellas instalaciones era bastante impresionante. Las máquinas de fundición a presión eran unos dispositivos grandes e imponentes, cada uno de los cuales estaba manejado por una sola persona. El molde se cerraba, se inyectaba aluminio líquido, el molde se abría y el operario sacaba la fundición de aluminio caliente del molde y la ponía en una cesta de metal grande.

Por la parte de arriba había un sistema de raíles por el cual un vehículo que transportaba un cubo enorme de aleación de aluminio fundido se desplazaba entre el horno y las máquinas de fundición individuales. Era impresionante.

Nuestro trabajo era programar un IBM System/7 para monitorizar el progreso de todas las máquinas de fundición. Contábamos el número de piezas que hacía cada máquina, cronometrábamos cada molde e informábamos de los desechos producidos. Eso se lograba mediante una red de área local enorme que conectaba las máquinas de fundición y varias estaciones de elaboración de informes en la planta con el System/7, que estaba arriba, en la sala de control, desde la que podía verse toda la planta a través de una ventana de observación.

IBM entró tarde en el juego de los miniordenadores. Habían apostado por los *mainframes*. Creo que pensaban que los miniordenadores iban a ser flor de un día. Si es así, se equivocaban. Al final, se dieron cuenta de que DEC y otras empresas de miniordenadores estaban dominando las aplicaciones de control de planta y querían un trozo del pastel, así que idearon el System/7.



Con un solo vistazo al panel de control, ya se sabía que era un dispositivo IBM. El System/7 tenía una arquitectura de 16 bits. La memoria era de estado sólido en vez de ser de núcleos. Me parece que el nuestro era 8K. Tenía ocho registros y un conjunto de instrucciones basado en registros muy simple (RISC).[3] Tenía un tiempo de ciclo del orden de un microsegundo. Creo que el nuestro tenía una unidad de disco interna pequeña.

También era físicamente grande para la época. Un PDP-8 o un M365 cabían debajo de un escritorio. El System/7 era del tamaño del congelador de un restaurante.

OMC envió a unos cuantos de nosotros al gran edificio de IBM en Chicago, junto a las torres Marina. Allí, durante cinco días, aprendimos el ensamblador System/7. Recuerdo que la clase parecía casi un chiste. El conjunto de instrucciones era trivial para aprender. Cinco días era demasiado tiempo.

En cualquier caso, volvimos con nuestros certificados y ya éramos programadores de System/7 con credenciales. Después de haber pasado un año editando código en terminales de vídeo, de pronto retrocedí al mundo de las tarjetas

3. *Reduced instruction set computer*, ordenador con conjunto de instrucciones reducido. Era lo opuesto a la tendencia de hacer las instrucciones de los ordenadores aún más complicadas. La idea era que una máquina RISC requería menos hardware y, por tanto, podía ser más barata y rápida.

perforadas. El System/7 no tenía compilador nativo. La compilación se hacía en el gran *mainframe* IBM 370 situado en el edificio de TI a casi un kilómetro de la planta, así que tuve que volver a las hojas de codificación y las perforadoras.

El código fuente se guardaba en archivos de disco en el 370. Editábamos esos archivos utilizando el mismo enfoque antiguo de edición de línea que había utilizado en A.S.C. Perforábamos directivas de edición en tarjetas, enviábamos estas tarjetas al 370 y, después, enviábamos trabajos de compilación al 370.

El 370 era un ordenador *mainframe* muy grande que estaba en una sala de ordenadores que nunca llegué a ver. Se trataba de una máquina orientada a lotes que solo podía hacer los trabajos de uno en uno.[4] Nuestros trabajos de edición y compilación esperaban en una cola de lotes, en el disco, durante horas. Una vez que la máquina encontraba tiempo para nuestros trabajos, transmitía el binario por una conexión de red privada al System/7, donde se almacenaba en el disco interno. El listado se enviaba a una impresora remota en la sala de control.

Así pues, los programadores íbamos y veníamos desde la planta al edificio con el ordenador. Perforábamos tarjetas y cargábamos nuestros trabajos en el edificio de TI y, después, regresábamos a la planta y esperábamos a que llegase la compilación. Dependiendo de lo ocupado que estuviese el 370, esa compilación podía llevar una hora o podía llevar un día. Lo cierto es que nunca lo sabíamos a ciencia cierta.

Una vez que llegaba el archivo binario y se almacenaba con seguridad en el disco interno del System/7, podíamos ejecutarlo y probarlo. El panel frontal era nuestro depurador. Tenía un modo de ejecución de paso único y podíamos ver los contenidos de los registros en las luces. Depurábamos, marcábamos nuestros listados y, después, volvíamos al edificio de TI para volver a empezar todo el proceso.

Yo lo odiaba. Odiaba las corbatas. Odiaba la burocracia. Odiaba la ineficiencia.

Odiaba la cultura. Lo odiaba todo tanto que no conseguía obligarme a llegar al trabajo a mi hora ni a tomarme los plazos en serio. Al final, me despidieron (y me lo merecía).

Pero, antes de ese día funesto, me pasaron dos cosas buenas mientras estaba en OMC. En junio de 1975, nació mi primera hija. Y, a principios de 1976, tuve un presentimiento de cuál sería mi futuro como consultor, autor y formador.

4. Bueno, a veces podía hacer dos, pero no de la manera en que usted y yo pensaríamos hoy.

Y, por supuesto, pese a las diferencias ideológicas en los diseños, el sistema funcionaba muy bien. El cliente y la empresa estaban satisfechos.

Los cuatro años de proyectos anteriores fueron una especie de revelación para mí. Estaba empezando a entender qué eran los principios del diseño de software. La vectorización inicial de los chips ROM del 8085 me planteó la idea de la capacidad para desarrollar e implementar de manera independiente. La pCCU y, más tarde, la DLU/DRU me hizo pensar en los distintos modos de procesos cooperantes. Y, por supuesto, el lenguaje C fue un lubricante significativo para esas ideas.

Esas ideas estaban a punto de unirse en un proyecto masivo que me consumiría durante varios años.

La empresa estaba creciendo y las oficinas que teníamos en Northbrook se habían quedado pequeñas. Era hora de construir nuestro propio edificio. Se hicieron plantas, se contrataron arquitectos y comenzó la construcción. Mientras tanto, nos trasladamos a unas instalaciones temporales en Wolf Road, en Wheeling, a solo unos kilómetros. Estuvimos allí un año.

Aprovechamos la oportunidad de la mudanza para cambiar el PDP-11/60 por un VAX 750. El VAX era una máquina mucho más rápida y potente. Tenía un megabyte de RAM y un tiempo de ciclo de 320 ns. ¡Era una bestia!

Lo instalamos en la sala de ordenadores y lo conectamos todo.

Ejecutar VMS era mucho mejor que ejecutar RSX 11-M. Todos nuestros compiladores y herramientas todavía funcionaban en modo PDP-11. Nuestras dificultades con el ensamblador de BSO se disiparon porque teníamos memoria de sobra y una máquina más rápida. ¡La vida era maravillosa!

EL APPLE II

Mientras tanto, apareció algo nuevo en la oficina de nuestro director financiero. Ahí, en su escritorio, había un Apple II, que utilizaba para ejecutar VisiCalc, la primera aplicación de hoja de cálculo.

Esa fue la primera vez que vi un ordenador personal empleado en un entorno empresarial. También fue la primera vez que vi un ordenador en el escritorio de una persona. Estaba convencido de que habría un ordenador en mi escritorio en un futuro cercano, pero no estaba seguro de cómo iba a justificarlo.

En los dos años siguientes, fueron apareciendo más y más Apple II en la oficina, pero siempre estaban en las mesas de la gente de negocios.

Contables, directores de ventas, directores de marketing... Todos ellos necesitaban poder utilizar hojas de cálculo. Todos tenían ordenadores. Me daban envidia.

PRODUCTOS NUEVOS

Las empresas que están creciendo necesitan productos nuevos. El director ejecutivo reunió a algunos de nosotros y nos dijo que pensásemos en qué nuevos productos podríamos producir y vender. Ahí estábamos, justo en medio de la revolución informática y la revolución de la telefonía. Las oportunidades eran ilimitadas. Mi jefe, Ken Finder, mi colega Jerry Fitzpatrick y yo fuimos elegidos para determinar una nueva dirección.

Para empezar, teníamos que espabilarnos con un montón de tecnologías, así que era hora de jugar. Dedicamos más o menos un año a crear prototipos de tecnologías de voz y a enredar con las nuevas unidades de disco ST-506 de 5,25 pulgadas y 5 MB de Seagate.

En un momento dado, pensamos en utilizar generadores de fonemas. Jerry montó de forma experimental un sencillo generador de fonemas y una interfaz telefónica y yo hice un programa en C 8085 que tomaba una frase del teclado, la descomponía en palabras, buscaba los fonemas para esa palabra y los enviaba al generador. También podía marcar un teléfono.

El 8085 extraía la biblioteca de fonemas del VAX utilizando un puerto serie y la cargaba en un árbol binario en RAM. Esa fue la primera vez que escribí un recorrido recursivo en árbol binario y me entusiasmó.

Una vez que estuvo listo, utilizamos el sistema para llamar a personas de la oficina y hacerles una serie de preguntas, como: "¿Quién fue el primer presidente de Estados Unidos?". La voz de los fonemas era muy robótica, pero, según nuestras investigaciones, era comprensible. Sin embargo, al final, optamos por una tecnología muy diferente.

Mientras tanto, yo era el administrador de sistema del VAX 750 y nos estábamos expandiendo con rapidez más allá de sus capacidades, así que planeé que en el edificio nuevo se instalase un VAX 780.

Cuando nos estábamos preparando para la mudanza, Ken, Jerry y yo concretamos nuestra visión para el nuevo producto. En el otoño de 1981, propusimos ese producto nuevo: el recepcionista electrónico (*Electronic Receptionist*, E. R.), el primer buzón de voz y sistema de gestión de llamadas digital del mundo.

Los programas de hojas de cálculo los crean programadores, pero las hojas de cálculo en sí las crean contables. Los motores de redes neuronales los crean programadores, pero las redes neuronales las crean ingenieros de redes neuronales. Las redes neuronales no pueden existir sin software, pero el software puede existir sin redes neuronales.

Por tanto, aunque las herramientas que hay detrás de las redes neuronales son algo en lo que los programadores estaremos muy implicados, las redes neuronales en sí tienen poco que ver con el futuro de la programación. Son solo una de las muchas aplicaciones de las que estaremos ocupándonos en el futuro.

Cuando contrata a un programador para escribir un programa, no piensa que va a "tener la esperanza"[2] de que el programa funcione. Contrata a un programador para que le proporcione un programa que genere resultados seguros. El recurso de un programador es el determinismo. Nosotros, los programadores, no trabajamos con esperanzas;[3] trabajamos con verdades binarias. Los sistemas que creamos son deterministas. Puede que otros utilicen las salidas deterministas de los sistemas que producimos como representación de la esperanza, pero esas salidas no dejan de ser deterministas. Trabajamos con hechos definidos sin ambigüedades. Utilizamos esos hechos para crear herramientas deterministas que ayudan a otros en su lucha con las nubes amorfas de la ambigüedad.

MODELOS DE LENGUAJE GRANDES

Cuando mezclamos una red neuronal pequeña[4] de 4 bits, un bonito algoritmo de cadena de Markov, el contenido de la Biblioteca del Congreso, todas las páginas de la web y todos los artículos de investigación de todas las bibliotecas universitarias, podemos obtener resultados bastante sorprendentes.

Sorprendentes, pero no necesariamente inteligentes.

O, quizá, debería decir que los resultados son inteligentes a veces, pero no creativos ni únicos. Y, si los resultados son inteligentes, no se debe a una inteligencia innata dentro del software del LLM. Más bien, cualquier inteligencia que perciba será necesariamente el resultado de la inteligencia de los escritos en los que se basa el LLM.

2. Hay muchas esperanzas puestas en los plazos y el presupuesto, pero hay una sola definición de "programa que funciona" (bueno... esperemos).
3. ¡Eso espero, desde luego!
4. Según alguna definición de "pequeña".

Verá, un LLM es un ejemplo excelente de GIGO (*garbage in, garbage out*, entra basura, sale basura). Si, por ejemplo, entrenase software de LLM utilizando los escritos de criminales, obtendría una salida muy criminal. Si lo entrenase con los escritos de maníacos genocidas, obtendría como salida mucha manía genocida. Y si lo entrena con los escritos de personas relativamente inteligentes, recibirá como salida una inteligencia relativa (a veces).

Pero en ningún caso recibirá nada único ni creativo.

Por ejemplo, pedí a un famoso LLM que dijese algo sabio. Su respuesta fue:

```
La vida es un viaje donde la belleza no solo está en llegar al destino, sino también
en saborear cada paso del camino. Abraza las lecciones, atesora los momentos y
encuentra la sabiduría en el tapiz de experiencias que dan forma a tu historia única.
```

Si no está poniendo los ojos en blanco, es probable que tenga menos de 30 años. ¿La vida es un viaje? ¿En serio? Bueno, vale, sí, pero esa vieja metáfora lleva muchos, muchos años utilizándose y ya se ha hecho un hueco en nuestras mentes (si tenemos más de 30 años).

Le pedí al mismo LLM que dijese algo único y dijo:

```
En la sinfonía de la existencia, cada individuo es una nota y contribuye a una melodía
única en la grandiosa composición del universo. Abraza la rareza de tu propia canción,
porque es en la armoniosa diversidad de todas las notas donde se revela la verdadera
belleza de la música de la vida.
```

Lo sé. Lo siento por sus músculos oculares. Se le deben quedar bastante doloridos después de eso. Pero bueno, haga una búsqueda en DuckDuckGo de "En la sinfonía de la existencia" y verá lo única que era esa afirmación.

Es bastante fácil convencernos de que los LLM no son inteligentes. Las respuestas a las dos preguntas anteriores no pasan la prueba de Turing. Ningún humano inteligente de más de 30 años habría respondido así a esas preguntas, a menos que estuviese haciendo un *sketch* cómico sobre los modelos de lenguaje grandes.

Eso no quiere decir que los LLM no sean útiles. Está claro que lo son. A menudo, son una alternativa mejor a un motor de búsqueda. Por ejemplo, le pregunté: "¿Quién escribió: 'En la sinfonía de la existencia, cada individuo es una nota'?". Y respondió:

```
Yo.
```

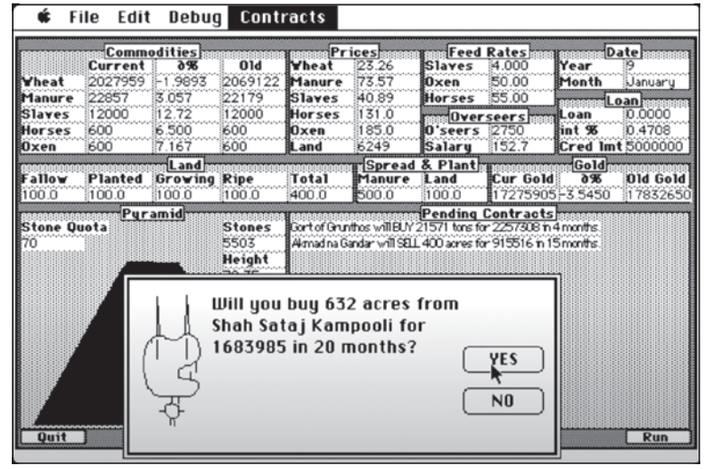
El primer modelo fue el 20, que se vendía por 11.800 dólares. Hubo muchos otros modelos, hasta llegar al 70, que soportaba 4 MB de memoria de estado sólido y tenía protección de memoria integrada, coma flotante y E/S muy rápida. Por lo general, estos sistemas se soportaban en memoria de disco y utilizaban memoria de cinta para las copias de seguridad.

***Perdidos en el espacio*:** Serie de televisión de ciencia ficción de 1965 basada ligeramente en *La familia Robinson*. Una familia se pierde en el espacio e intenta desesperadamente volver a casa. El robot, construido por los mismos ingenieros que crearon al robot Robby de *Planeta prohibido*, se convirtió en un icono por decir "Peligro, Will Robinson" y "No tiene sentido" mientras agitaba sus brazos corrugados de forma aleatoria.

Perforadora IBM 026: Anunciada en 1949, estas perforadoras de tarjetas de tamaño escritorio fueron las principales perforadoras controladas con teclado durante más de 20 años. El teclado incluía mayúsculas, números y algunos signos de puntuación. Las tarjetas se extraían de forma automática de un alimentador de entrada y a través del mecanismo perforador controlado por el teclado y, después, se apilaban en un receptor de salida.

Pharaoh: Mientras estaba en Teradyne en los setenta, había un juego escrito ALCOM llamado Pharaoh. Lo modifiqué enormemente y pasé mucho tiempo jugando. En 1987, decidí escribir el juego otra vez, en C, para mi Mac 128. Cuando

estuvo completo, lo subí a CompuServe o algo de eso. Circuló por ahí. La mayoría de la gente lo odiaba. A algunos les gustaba. Puede descargarlo en `www.macintoshrepository.org/5230-pharaoh`. En el momento de escribir esto, había una valoración muy divertida de 2018 disponible en: Tanara Kuranov (Gamer Mouse). "Gamer Mouse-Pharaoh Review-Macintosh". Publicado en YouTube el 20 de febrero de 2018.



Planeta prohibido: Película de 1956 (y mi película favorita de todos los tiempos). El robot Robby es un personaje importante y tiene la personalidad de un mayordomo inglés.

PostScript: Inventado en 1984 por Adobe, PostScript es un llamado "lenguaje de descripción de páginas". Basado en Forth, este lenguaje era la manera común de enviar trabajos de impresión desde un ordenador a una impresora láser a finales de los ochenta y los noventa. El ordenador convertía el trabajo de impresión en un programa PostScript y lo enviaba a la impresora. La impresora ejecutaba ese programa, lo que hacía que la impresora imprimiese páginas. Con el tiempo, PDF sustituyó a PostScript.

Proceso unificado racional (RUP): Incentivados por el *boom* de las empresas puntocom y financiados por Rational Inc., Grady Booch, Ivar Jacobson y Jim Rumbaugh (Los Tres Amigos) colaboraron para crear un proceso de desarrollo de software rico y diverso. Rational empezó a comercializar la idea en 1996. Al final, RUP se vio sobrepasado por el movimiento ágil y, sobre todo, por Scrum y la programación extrema.

El viaje de la programación y sus pioneros: desde el nacimiento del código al crecimiento de la IA

En *Nosotros, los programadores*, la leyenda del software Robert C. Martin ("Uncle Bob") se sumerge en el mundo de la programación, explorando la vida de los pioneros revolucionarios que crearon los cimientos de la informática moderna. Desde Charles Babbage y Ada Lovelace a Alan Turing, Grace Hopper y Dennis Ritchie, Martin pone el foco sobre las figuras cuyo brillo y perseverancia cambiaron el mundo.

Esta narración repleta de recuerdos ofrece una historia humana rica llena de perspectivas técnicas para desarrolladores y examina las innovaciones en la programación que dieron forma a la informática a nivel de bits y de bytes. Al conectar estos logros técnicos con las historias humanas que hay tras ellos, Martin ofrece a los lectores una visión poco habitual de las luchas y los triunfos de las personas que hicieron posible la tecnología moderna. Depresión, fracaso y ridículo, estos pioneros se enfrentaron a todo eso y sus vidas se entrelazan con la evolución de la propia computación a medida que el campo evolucionó desde sus orígenes humildes a las inteligencias artificiales basadas en la nube de la actualidad. Con el crecimiento de la IA, Martin también explora cómo esta tecnología está transformando el futuro de la programación y los desafíos éticos que surgen con ella.

Entre los temas destacados se incluyen:

- La comprensión de las raíces de la programación y el modo en que dieron forma al paisaje tecnológico de hoy.
- El lado humano de los pioneros de la programación, lo que los impulsaba y a lo que se enfrentaron.
- Innovaciones clave en la programación, desde los primeros días de los ensambladores al surgimiento de los lenguajes orientados a objetos.
- El papel crucial que jugó la Segunda Guerra Mundial para hacer avanzar la informática.
- Observaciones y predicciones sobre las consideraciones éticas que rodean la IA y el futuro de la programación.

Para programadores, codificadores y cualquiera que esté fascinado por la intersección entre las personas y las máquinas, esta guía sobre la historia, la humanidad y la tecnología detrás del código que mueve el mundo actual es una lectura fascinante y esencial.

Robert C. Martin ("Uncle Bob") es programador y experto en desarrollo de software desde 1970. Es el fundador de Uncle Bob Consulting, LLC y cofundador, junto a su hijo Micah Martin, de Clean Coders LLC. Martin ha publicado docenas de artículos en varias revistas especializadas y es orador habitual en conferencias y ferias internacionales. Es autor de muchos libros, incluidos *Agile Software Development: Principles, Patterns, and Practices; UML for Java Programmers; Código limpio; El limpiador de código; Arquitectura limpia; Desarrollo ágil esencial; La artesanía del código limpio* y *Diseño funcional*. Martin trabajó durante tres años como jefe de redacción de *C++ Report* y fue el primer presidente de la Alianza Ágil.

ANAYA MULTIMEDIA
www.anayamultimedia.es

ISBN 978-84-415-5201-2
2315274
9 788441 552012