

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/373708742>

# Principios de Programación

Book · September 2023

CITATIONS

0

READS

1,538

1 author:



**Camilo Chacón Sartori**

Spanish National Research Council

20 PUBLICATIONS 36 CITATIONS

SEE PROFILE

¿Cuáles son los principios subyacentes a toda herramienta en programación?

Si quiere conocer los ocho principios, técnicos y conductuales, que dan respuesta a esta pregunta, ha llegado al libro indicado.

En una época donde cada día surgen nuevas tecnologías, el beneficio de conocer conceptos transversales a todas ellas no solo es imprescindible sino necesario.

Además, con la llegada de sofisticadas aplicaciones de inteligencia artificial, la pregunta ya no reside en qué herramienta aprender, sino en qué tienen en común para poder integrarlas.

Gracias a la lectura de este libro, descubrirá los cinco tomos que lo componen y que dan soporte a la nueva forma de entender la programación.

- Tomo I: Aprenderá los fundamentos básicos de las matemáticas y de la programación.
- Tomo II: Conocerá los principios de programación
- Tomo III: Dispondrá de una introducción histórica y práctica a los diversos sistemas de la computación, como los lenguajes de programación, los sistemas operativos, las bases de datos, los sistemas distribuidos y la inteligencia artificial.
- Tomo IV: Analizará el diálogo que presenta los desafíos de la ingeniería de software.
- Tomo V: Disfrutará de reflexiones y consejos para crecer como profesional.

No pierda la oportunidad de iniciar el camino que le propone este libro, donde irá desde historia del campo a la programación en sí misma, a la vez que le suscitará nuevas ideas que impulsarán su carrera como programador.

**Camilo Chacón Sartori** es doctorante en el Instituto de Investigación en Inteligencia Artificial (IIIA-CSIC) y en la Universidad Autónoma de Barcelona. Obtuvo su grado y máster en ingeniería en informática con distinción máxima. Ha publicado dos libros: *Computación y programación funcional* y *Mentes geniales. La vida y obra de 12 grandes informáticos*, ambos por la editorial Marcombo. Su principal proyecto «Había una vez un algoritmo» es un pódcast y un newsletter donde reflexiona sobre temas técnicos, científicos y filosóficos concernientes a la informática.



## PRINCIPIOS DE PROGRAMACIÓN



# PRINCIPIOS DE PROGRAMACIÓN

CAMILO CHACÓN SARTORI



# **Principios de programación**

Camilo Chacón Sartori



## ÍNDICE DE CONTENIDOS

<b>NOTA AL LECTOR .....</b>	<b>21</b>
<b>AGRADECIMIENTOS.....</b>	<b>23</b>
<b>INTRODUCCIÓN.....</b>	<b>25</b>
<b>MAPA DE LECTURA.....</b>	<b>31</b>
<b>TOMO I – FUNDAMENTOS.....</b>	<b>33</b>
<b>CAPÍTULO 1 LOS FUNDAMENTOS MATEMÁTICOS .....</b>	<b>39</b>
1.1.    INTRODUCCIÓN.....	39
1.2.    LÓGICA.....	42
1.2.1. <i>Lógica proposicional</i> .....	42
1.3.    CONJUNTOS.....	55
1.3.1. <i>Notación</i> .....	56
1.3.2. <i>Subconjunto</i> .....	57
1.4.    CONCLUSIÓN.....	58
1.4.1. <i>Lecturas sugeridas</i> .....	59
<b>CAPÍTULO 2 CONCEPTOS ELEMENTALES .....</b>	<b>61</b>
2.1.    INTRODUCCIÓN.....	61
2.2.    SECUENCIALES.....	62
2.2.1. <i>Variables</i> .....	62
2.2.2. <i>Condicional</i> .....	73
2.2.3. <i>Descomposición de conectivas</i> .....	76
2.2.4. <i>Bucle</i> .....	80
2.3.    FUNCIONALES.....	87
2.3.1. <i>Función</i> .....	87
2.3.2. <i>Recursión</i> .....	96
2.4.    SECUENCIAL VS. RECURSIVIDAD .....	104
2.5.    CONCLUSIÓN.....	106

## **CAPÍTULO 3 ALGORITMOS Y ESTRUCTURAS DE DATOS ..... 107**

3.1.	INTRODUCCIÓN.....	107
3.2.	ALGORITMOS COMO TECNOLOGÍA .....	108
3.2.1.	<i>Análisis de algoritmos .....</i>	<i>109</i>
3.2.2.	<i>Clasificación.....</i>	<i>110</i>
3.3.	ESTRUCTURAS DE DATOS COMO COMPLEMENTO .....	119
3.3.1.	<i>Arreglos.....</i>	<i>119</i>
3.3.2.	<i>Listas enlazadas.....</i>	<i>122</i>
3.4.	OTRAS.....	125
3.5.	CONCLUSIÓN.....	125

## **CAPÍTULO 4 PENSAMIENTO COMPUTACIONAL ..... 127**

4.1.	INTRODUCCIÓN.....	127
4.2.	ORIGEN.....	129
4.3.	LOS CINCO ASPECTOS BÁSICOS DEL PENSAMIENTO COMPUTACIONAL.....	131
4.3.1.	<i>Modularidad.....</i>	<i>131</i>
4.3.2.	<i>Estructura de datos .....</i>	<i>133</i>
4.3.3.	<i>Encapsulación.....</i>	<i>134</i>
4.3.4.	<i>Estructuras de control.....</i>	<i>135</i>
4.3.5.	<i>Recursión .....</i>	<i>137</i>
4.4.	OTROS GRUPOS DE CONCEPTOS DEL PENSAMIENTO COMPUTACIONAL .....	139
4.5.	CONCLUSIÓN.....	139
4.5.1.	<i>Lecturas sugeridas.....</i>	<i>139</i>

## **TOMO II – PRINCIPIOS.....141**

## **PRINCIPIOS TÉCNICOS .....151**

## **CAPÍTULO 1 DISEÑO ..... 155**

1.1.	INTRODUCCIÓN .....	155
1.2.	LOS PRERREQUISITOS DE UN DISEÑO .....	156
1.2.1.	<i>Objetivo del diseño de software .....</i>	<i>157</i>
1.3.	CONCEPTOS GENERALES.....	160
1.3.1.	<i>Descomposición y composición.....</i>	<i>160</i>

1.3.2.	<i>Refactorización</i> .....	172
1.3.3.	<i>Patrones de diseño</i> .....	178
<b>1.4.</b>	<b>COMUNICACIÓN ENTRE PERSONAS</b> .....	187
1.4.1.	<i>Diseño e implementación</i> .....	188
1.4.2.	<i>Lenguajes de modelado</i> .....	190
1.4.3.	<i>El futuro del diseño de software</i> .....	192
<b>1.5.</b>	<b>CONCLUSIÓN</b> .....	194
1.5.1.	<i>Lecturas sugeridas</i> .....	194
<b>CAPÍTULO 2</b>	<b>ESTADO</b> .....	<b>197</b>
<b>2.1.</b>	<b>INTRODUCCIÓN</b> .....	197
<b>2.2.</b>	<b>ASIGNACIÓN</b> .....	198
2.2.1.	<i>Flujo de estados</i> .....	201
2.2.2.	<i>Lenguaje ensamblador</i> .....	202
<b>2.3.</b>	<b>ORDEN Y TIEMPO</b> .....	217
2.3.1.	<i>Autómata finito (determinista y no-determinista)</i> .....	217
2.3.2.	<i>Autómata celular</i> .....	223
<b>2.4.</b>	<b>CONCLUSIÓN</b> .....	227
2.4.1.	<i>Lecturas sugeridas</i> .....	227
<b>CAPÍTULO 3</b>	<b>RECURSO</b> .....	<b>229</b>
<b>3.1.</b>	<b>INTRODUCCIÓN</b> .....	229
<b>3.2.</b>	<b>COMPLEJIDAD ALGORÍTMICA</b> .....	230
3.2.1.	<i>Lineal</i> .....	233
3.2.2.	<i>Cuadrática</i> .....	236
3.2.3.	<i>Exponencial</i> .....	237
3.2.4.	<i>Logarítmica</i> .....	239
<b>3.3.</b>	<b>LIMITACIONES</b> .....	240
<b>3.4.</b>	<b>AVANCES DEL HARDWARE</b> .....	242
<b>3.5.</b>	<b>MÉTRICAS</b> .....	243
3.5.1.	<i>Particulares</i> .....	244
3.5.2.	<i>Generales</i> .....	245

3.6.	CONCLUSIÓN.....	246
3.6.1.	<i>Lecturas sugeridas</i> .....	247
<b>CAPÍTULO 4 COORDINACIÓN.....</b>		<b>249</b>
4.1.	INTRODUCCIÓN .....	249
4.2.	COMUNICACIÓN: ORDEN Y TIEMPO .....	251
4.2.1.	<i>Exclusión mutua</i> .....	251
4.3.	CONCURRENCIA Y PARALELISMO.....	255
4.3.1.	<i>Ley de Amdahl</i> .....	257
4.3.2.	<i>¿Cuándo paralelizar?</i> .....	258
4.3.3.	<i>GPU</i> .....	259
4.4.	COMPUTACIÓN DISTRIBUIDA .....	259
4.4.1.	<i>MapReduce</i> .....	260
4.4.2.	<i>Paso de mensajes</i> .....	261
4.4.3.	<i>Llamada a procedimiento remoto</i> .....	262
4.5.	CONCLUSIÓN.....	263
4.5.1.	<i>Lecturas sugeridas</i> .....	264
<b>CAPÍTULO 5 TRANSFORMACIÓN.....</b>		<b>265</b>
5.1.	INTRODUCCIÓN .....	265
5.2.	REPRESENTACIONES .....	269
5.2.1.	<i>Funciones</i> .....	269
5.2.2.	<i>Formato de fichero</i> .....	272
5.2.3.	<i>Sistema de numeración</i> .....	278
5.2.4.	<i>Serialización</i> .....	280
5.2.5.	<i>Computación reversible</i> .....	286
5.3.	OTROS TIPOS DE TRANSFORMACIONES .....	287
5.4.	CONCLUSIÓN.....	288
5.4.1.	<i>Lecturas sugeridas</i> .....	289
<b>PRINCIPIOS CONDUCTUALES .....</b>		<b>291</b>
<b>CAPÍTULO 6 DOCUMENTACIÓN .....</b>		<b>293</b>
6.1.	INTRODUCCIÓN .....	293

6.2.	¿DOCUMENTA, POR FAVOR?.....	294
6.3.	LOS PROBLEMAS DE DOCUMENTAR UN SOFTWARE .....	296
6.4.	ESTRATEGIAS PARA DOCUMENTAR.....	298
6.4.1.	<i>El código como documentación .....</i>	<i>298</i>
6.4.2.	<i>Documentación viva .....</i>	<i>306</i>
6.5.	CONCLUSIÓN.....	307
6.5.1.	<i>Lecturas sugeridas .....</i>	<i>308</i>
<b>CAPÍTULO 7 EVALUACIÓN.....</b>		<b>309</b>
7.1.	INTRODUCCIÓN .....	309
7.2.	PRUEBAS DINÁMICAS .....	310
7.2.1.	<i>Funcionales.....</i>	<i>311</i>
7.2.2.	<i>No funcionales .....</i>	<i>312</i>
7.3.	PRUEBAS ESTÁTICAS .....	316
7.3.1.	<i>Inspección de los requerimientos de software .....</i>	<i>317</i>
7.3.2.	<i>Análisis estático.....</i>	<i>318</i>
7.4.	REPLICACIÓN Y CONTRASTACIÓN.....	322
7.4.1.	<i>¿Cómo replicar?.....</i>	<i>323</i>
7.4.2.	<i>¿Cómo contrastar? .....</i>	<i>324</i>
7.5.	CONCLUSIÓN.....	325
7.5.1.	<i>Lecturas sugeridas .....</i>	<i>326</i>
<b>CAPÍTULO 8 ETHOS.....</b>		<b>327</b>
8.1.	INTRODUCCIÓN .....	327
8.2.	LA RESPONSABILIDAD FRENTE A LA INTELIGENCIA ARTIFICIAL .....	328
8.2.1.	<i>IA y robótica .....</i>	<i>329</i>
8.2.2.	<i>IA y sexualidad.....</i>	<i>329</i>
8.2.3.	<i>Principio ético.....</i>	<i>330</i>
8.3.	¿UN PROGRAMADOR TIENE RESPONSABILIDAD?.....	331
8.4.	¿QUÉ ES SER UN BUEN PROGRAMADOR? .....	332
8.4.1.	<i>Moralidad y programación .....</i>	<i>332</i>



8.5.	CONCLUSIÓN.....	334
8.5.1.	<i>Lecturas sugeridas</i> .....	334
<b>TOMO III – SISTEMAS .....</b>		<b>335</b>
<b>CAPÍTULO 1 LENGUAJES DE PROGRAMACIÓN.....</b>		<b>341</b>
1.1.	INTRODUCCIÓN.....	341
1.2.	BREVE HISTORIA .....	344
1.3.	COMPONENTES DE UN LENGUAJE DE PROGRAMACIÓN .....	350
1.3.1.	<i>Sintaxis</i> .....	350
1.3.2.	<i>Semántica</i> .....	351
1.4.	CATEGORÍAS DE LENGUAJES .....	352
1.4.1.	<i>Generales</i> .....	352
1.4.2.	<i>Dominio específico</i> .....	353
1.5.	ESTILOS DE LENGUAJES .....	354
1.5.1.	<i>Funcional</i> .....	354
1.5.2.	<i>Imperativo</i> .....	356
1.5.3.	<i>Orientado a objeto</i> .....	356
1.5.4.	<i>Otros</i> .....	357
1.6.	DISEÑO E IMPLEMENTACIÓN .....	360
1.7.	FUTURO .....	363
1.8.	CONCLUSIÓN .....	365
1.8.1.	<i>Lecturas recomendadas</i> .....	365
<b>CAPÍTULO 2 SISTEMAS OPERATIVOS.....</b>		<b>367</b>
2.1.	INTRODUCCIÓN.....	367
2.2.	BREVE HISTORIA .....	368
2.3.	CONCEPTOS FUNDAMENTALES .....	377
2.3.1.	<i>Virtualización</i> .....	377
2.3.2.	<i>Concurrencia</i> .....	381
2.3.3.	<i>Persistencia</i> .....	384
2.4.	FUTURO .....	386

2.5.	CONCLUSIÓN .....	387
2.5.1.	<i>Lecturas recomendadas</i> .....	387
<b>CAPÍTULO 3 BASE DE DATOS .....</b>		<b>389</b>
3.1.	INTRODUCCIÓN.....	389
3.2.	BREVE HISTORIA .....	391
3.3.	BASES DE DATOS RELACIONALES.....	395
3.3.1.	<i>Modelo relacional</i> .....	397
3.4.	BASES DE DATOS NO RELACIONALES (NoSQL) .....	400
3.4.1.	<i>Documentos</i> .....	401
3.4.2.	<i>Clave-valor</i> .....	401
3.4.3.	<i>Grafos</i> .....	402
3.4.4.	<i>Vectores</i> .....	402
3.5.	FUTURO .....	402
3.6.	CONCLUSIÓN .....	404
3.6.1.	<i>Lecturas recomendadas</i> .....	404
<b>CAPÍTULO 4 SISTEMAS DISTRIBUIDOS .....</b>		<b>405</b>
4.1.	INTRODUCCIÓN.....	405
4.2.	BREVE HISTORIA .....	406
4.3.	ARQUITECTURAS DE COMPUTACIÓN DISTRIBUIDA .....	410
4.3.1.	<i>Basada en capas</i> .....	410
4.3.2.	<i>Orientada a servicios</i> .....	412
4.3.3.	<i>Publicador-suscriptor</i> .....	413
4.4.	CARACTERÍSTICAS.....	414
4.4.1.	<i>Procesos</i> .....	415
4.4.2.	<i>Comunicación</i> .....	416
4.4.3.	<i>Coordinación</i> .....	421
4.4.4.	<i>Consistencia y replicación</i> .....	421
4.4.5.	<i>Tolerancia a fallas</i> .....	422
4.4.6.	<i>Seguridad</i> .....	422
4.5.	FUTURO .....	423

4.6.	CONCLUSIÓN .....	424
4.6.1.	<i>Lecturas recomendadas</i> .....	424
<b>CAPÍTULO 5 INTELIGENCIA ARTIFICIAL.....</b>		<b>425</b>
5.1.	INTRODUCCIÓN.....	425
5.2.	UNA BREVE HISTORIA.....	426
5.3.	ENFOQUE SIMBÓLICO.....	442
5.4.	ENFOQUE PROBABILISTA .....	443
5.4.1.	<i>Aprendizaje automático (Machine Learning)</i> .....	443
5.4.2.	<i>Aprendizaje profundo (Deep Learning)</i> .....	444
5.5.	FUTURO .....	446
5.6.	CONCLUSIÓN .....	447
5.6.1.	<i>Lecturas recomendadas</i> .....	447
<b>TOMO IV – SOMOS HUMANOS .....</b>		<b>449</b>
<b>CAPÍTULO 1 BUENAS PRÁCTICAS .....</b>		<b>455</b>
<b>CAPÍTULO 2 INGENIERÍA DE SOFTWARE.....</b>		<b>461</b>
<b>CAPÍTULO 3 TIPOS DE SOFTWARE .....</b>		<b>467</b>
<b>TOMO V – BUENA VIDA.....</b>		<b>473</b>
<b>CAPÍTULO 1 APRENDIZAJE DE PROGRAMACIÓN .....</b>		<b>477</b>
1.1.	INTRODUCCIÓN.....	477
1.2.	HERRAMIENTAS .....	478
1.2.1.	<i>El terminal es su mejor amigo</i> .....	478
1.2.2.	<i>La importancia de depurar</i> .....	481
1.2.3.	<i>Cuando el control de versiones es útil</i> .....	482
1.2.4.	<i>Manipular texto</i> .....	485
1.2.5.	<i>Reinventar la rueda</i> .....	487
1.2.6.	<i>Use generadores de código</i> .....	489
1.3.	BUENAS PRÁCTICAS .....	490
1.3.1.	<i>Sobre leer código</i> .....	491
1.3.2.	<i>Cree una historia de su código</i> .....	492

1.3.3.	<i>Sobre escribir código.....</i>	494
<b>1.4.</b>	<b>AVANZAR EN SU CARRERA .....</b>	<b>495</b>
1.4.1.	<i>Averigüe qué tipo de programador es.....</i>	495
1.4.2.	<i>Cuando enseñar también significa aprender.....</i>	501
1.4.3.	<i>Construya una comunidad.....</i>	504
<b>1.5.</b>	<b>FILOSÓFICOS .....</b>	<b>510</b>
1.5.1.	<i>Pensar antes de programar.....</i>	510
1.5.2.	<i>Cuide sus palabras .....</i>	513
1.5.3.	<i>Sea analítico .....</i>	514
1.5.4.	<i>Procure ser un generalista.....</i>	515
<b>1.6.</b>	<b>CONCLUSIÓN .....</b>	<b>518</b>
1.6.1.	<i>Lecturas sugeridas.....</i>	518
<b>CAPÍTULO 2</b>	<b>VIDA PERSONAL .....</b>	<b>521</b>
<b>2.1.</b>	<b>INTRODUCCIÓN.....</b>	<b>521</b>
<b>2.2.</b>	<b>CIENCIAS.....</b>	<b>522</b>
2.2.1.	<i>La programación no son matemáticas, ¡pero estas ayudan! .....</i>	522
2.2.2.	<i>Las ciencias como un faro en la oscuridad .....</i>	523
<b>2.3.</b>	<b>ARTES Y HUMANIDADES.....</b>	<b>525</b>
2.3.1.	<i>La música y la programación.....</i>	526
2.3.2.	<i>Dibujo como inspiración.....</i>	527
<b>2.4.</b>	<b>OCIO .....</b>	<b>530</b>
2.4.1.	<i>Pasear, mirar y escuchar .....</i>	530
2.4.2.	<i>No hacer nada.....</i>	531
<b>2.5.</b>	<b>CONCLUSIÓN .....</b>	<b>532</b>
2.5.1.	<i>Lecturas sugeridas.....</i>	532
<b>REFLEXIONES FINALES .....</b>		<b>535</b>
<b>ÍNDICE DE NOMBRES .....</b>		<b>547</b>
<b>ÍNDICE TEMÁTICO .....</b>		<b>551</b>

## NOTA AL LECTOR

Hace años, acaso por una cierta ingenuidad y entusiasmo propio de la juventud, quise escribir un libro que concentrara todo lo que sé de la programación, una forma de restituir la mano que me tendió cuando mi único camino era la informática. Desde entonces ha pasado un tiempo y ahora usted tiene en su poder esta promesa que hice: un libro que yo hubiera querido leer y que no encontré; un libro que intentara aunar gran parte de los conceptos presentes en la programación; un libro que, además de ser una apuesta, es un intento de romper la clásica estructura de un manual de programación, pues este es un libro generalista, no se enfoca en nada en particular, ni en un área ni en una herramienta, pero no pierde de vista que la parte práctica es la culminación de la programación.

Antes bien: es necesario recorrer un camino lleno de historias, de conceptos y de algoritmos, e igualmente, por qué no decirlo, de sueños y de pensar cómo podría ser la programación como campo de estudio, algo que, en la actualidad, no es.

Así pues, la obra que tiene en su poder fue pensada como el ingreso a una biblioteca pública dedicada a la informática, donde en algunas estanterías hallará libros de historia del campo, en otras encontrará libros de ficción sobre programación, caminando por el pasillo de al lado podrá hojear libros técnicos... Y así, mientras

recorre esta biblioteca —y es la idea que tengo— le irán surgiendo nuevas ideas y lecturas. ¡Más que un fin es el inicio del camino!

¡Bienvenido/a a su casa!

Espero que disfrute de este libro de programación que, a su vez, reflexiona sobre ella.

Castro, Chile<sup>1</sup>

---

<sup>1</sup> Este libro fue escrito casi en su totalidad en Barcelona (España) y fue terminado en Castro (Chile).

## Referencias

- Aghajani, E., Nagy, C., Vega-Marquez, O. L., Linares-Vasquez, M., Moreno, L., Bavota, G.y Lanza, M. (2019). Software Documentation Issues Unveiled. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*.
- Agin, W. E. (2019). «A History of Artificial Intelligence». En *Law of Artificial Intelligence and Smart Machines*.
- Allamanis, M., Barr, E. T., Bird, C.y Sutton, C. (2014). «Learning natural coding conventions». (págs. 281-293). Association for Computing Machinery.
- Allamanis, M., Barr, E. T., Bird, C.y Sutton, C. (2015). «Suggesting accurate method and class names». *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (págs. 38-49). Association for Computing Machinery.
- Angles, R.y Gutierrez, C. (2008). «Survey of graph database models». *ACM Comput. Surv.*
- Avram, A. (2016). *Details on How Linux Runs Natively on Windows* . Obtenido de InfoQ: <https://www.infoq.com/news/2016/04/linux-windows/>
- Beck, K. a. (2001). *Manifiesto por el Desarrollo Ágil de Software*. Obtenido de <https://agilemanifesto.org/iso/es/manifesto.html>
- Bhattacharjee, K., Naskar, N., Roy, S.y Das, S. (2020). «A survey of cellular automata: types, dynamics, non-uniformity and applications». *Nat. Comput.*, 433-461.
- Biancuzzi, F.y Warden, S. (2009). *Masterminds of programming: Inspiring conversations with creators of major programming languages*. O'Reilly Media.

- Bieman, J. M.y Kang, B.-K. (1995). «Cohesion and reuse in an object-oriented system». *SIGSOFT Softw. Eng. Notes*, 259-262.
- Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., . . . Goodman, N. D. (2018). «Pyro: Deep Universal Probabilistic Programming». *arXiv*.
- Böhm, C.y Jacopini, G. (1966). «Flow diagrams, Turing machines and languages with only two formation rules». *Communications of the ACM*, 366–371.
- Bonissone, P.y Johnson, J. H. (1984). «DELTA: An Expert System for Diesel Electric Locomotive Repair».
- Bqa, R., Shakeel, T.y Khan, Y. D. (2019). «A Pedagogical Approach towards Theory of Computation». *Proceedings of the 2019 8th International Conference on Educational and Information Technology*, 192-197.
- Bruderer, H. (2022). «Was Ada Lovelace Actually the First Programmer? ». *Communications of the ACM*.
- Budgen, D. (2003). *Software Design*. Pearson Education.
- Campbell, M., Hoane Jr., A. J.y Hsu, F.-h. (2002). «Deep Blue». *Artificial Intelligence* .
- Chacón Sartori, C. (2021). *Computación y programación funcional*. Marcombo.
- Chacón Sartori, C. (2021). *Mentes geniales. La vida y obra de 12 grandes informáticos*. Marcombo.
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde, H., Kaplan, J., . . . Zaremba, W. (2021). «Evaluating Large Language Models Trained on Code». *ArXiv*.
- Coulouris, G., Dollimore, J., Kindberg, T.y Blair, G. (2012). *Distributed Systems. Concepts and Design*.
- Cunningham, W. (1992). *The WyCash Portfolio Management System*.
- Curtis, B., Krasner, H.y Iscoe, N. (1988). «A field study of the software design process for large systems». *Commun. ACM*, 1268-1287.



- Dahl, O. J., Dijkstra, E. W. y Hoare, C. A. (1972). *Structured Programming*. Academic Press Ltd.
- Daylight, E. G. (2011). «Dijkstra's Rallying Cry for Generalization: The Advent of the Recursive Procedure, late 1950s — early 1960s». *Comput. J.*, 1756-1772.
- Daylight, E. G. (2012). *The Dawn of Software Engineering: From Turing to Dijkstra*. Lonely Scholar.
- Denning, P. J. y Tedre, M. (2021). «Computational Thinking: A Disciplinary Perspective». *Informatics in Education*.
- Devlin, K. (2020). «The Ethics of the Artificial Lover». En S. M. Liao, *Ethics of Artificial Intelligence*. Oxford University Press.
- Dey, B., Halder, S., Gendt, S. y Meert, W. (2022). «Code Generation Using Machine Learning: A Systematic Review». *IEEE Access*.
- Díaz, J. y Torras, C. (2012). «A personal account of Turing's imprint on the development of computer science». *Comput. Sci. Rev.*
- Dijkstra, E. W. (1975). *How do we tell truths that might hurt?* Obtenido de <https://www.cs.virginia.edu/~evans/cs655/readings/ewd498.html>
- Dyakonov, M. I. (2020). *Will We Ever Have a Quantum Computer?* Springer.
- Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M. y Zander, C. (2006). «Can graduating students design software systems? » *Proceedings of the 37th SIGCSE technical symposium on Computer science education*. Association for Computing Machinery.
- Esteero, R., Khan, M., Mohamed, M., Zhang, L. Y. y Zingaro, D. (2018). «Recursion or Iteration: Does it Matter What Students Choose? » *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (págs. 1011-1016). Association for Computing Machinery.
- Fowler, M. (2018). *Refactoring*. Addison-Wesley.

- Gamma, E. a. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.
- Goldreich, O. (2008). *Computational complexity. A conceptual perspective*. Cambridge University Press.
- H. Arpaci-Dusseau, R.y C. Arpaci-Dusseau, A. (2014). *Operating Systems: Three Easy Pieces*.
- Haenlein, M.y Kaplan, A. (2019). «A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence». *Berkeley Haas*.
- Havelund, K., Tegeler, T., Smyth, S.y Steffen, B. (2022). «Discussing the Future Role of Documentation in the Context of Modern Software Engineering (ISO/LA 2022 Track Introduction)». *Leveraging Applications of Formal Methods, Verification and Validation. Software Engineering*, 3-9.
- Hellerstein, J. M., Stonebraker, M.y Hamilton, J. (2007). «Architecture of a Database System». *Now Publishers Inc*.
- Hidary, J. D. (2019). *Quantum Computing: An Applied Approach*.
- Hoare, C. (1962). Quicksort. *The Computer Journal*, 10-16.
- Hoare, T. (1974). Hints on programming language design. En C. Bunyan, *State of the Art Report 20: Computer Systems Reliability* (págs. 505–534). Pergamon/Infotech.
- Holton, W. C. (2021). quantum computer. *Encyclopedia Britannica*.
- Høst, E. W.y Østvold, B. M. (2009). «Debugging Method Names». *Springer Berlin Heidelberg*, 294-317.
- Jerzyk, M. (2022). «Code Smells: A Comprehensive Online Catalog and Taxonomy». Springer.
- Jiayuan Mao, C. G. (2019). «The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision». *arXiv*.

- Kari, L.y Rozenberg, G. (2008). «The Many Facets of Natural Computing». *Communications of the ACM*.
- Kernighan, B. W.y Pike, R. (1999). *The Practice of Programming*. Addison-Wesley.
- Klein, G. (2019). Intuición. En J. Brockman, *Las mejores decisiones*. Crítica.
- Knuth, D. (1989). *Notes on the Errors of TeX*. Obtenido de <https://tug.org/TUGboat/tb10-4/tb26knut.pdf>
- Langston, J. (2021). *From conversation to code: Microsoft introduces its first product features powered by GPT-3*. Obtenido de <https://blogs.microsoft.com/ai/from-conversation-to-code-microsoft-introduces-its-first-product-features-powered-by-gpt-3/>
- LaPierre, R. (2022). *Introduction to Quantum Computing*. Springer.
- Ledin, J. (2022). *Modern Computer Architecture and Organization: Learn x86, ARM, and RISC-V architectures and the design of smartphones, PCs, and cloud servers*. Packt Publishing.
- Li, G., Liu, H.y Nyamawe, A. S. (2021). «A Survey on Renamings of Software Entities». *ACM Comput. Surv.*, 1-38.
- Li, G., Zhou, X.y Cao, L. (2021). «AI Meets Database: AI4DB and DB4AI». *Proceedings of the 2021 International Conference on Management of Data*.
- Lindsay, D., Gill, S. S., Smirnova, D.y Garraghan, P. (2023). «The evolution of distributed computing systems: From fundamentals to new frontiers». *Springer*.
- Liu, K., Kim, D., Bissyande, T. F., Kim, T., Kim, K., Koyuncu, A., . . . Le Traon, Y. (2019). «Learning to spot and refactor inconsistent method names». *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE.

- Luo, J., Zhang, J., Huang, Z., Xu, Y. y Sun, C. (2022). «Toward an accurate method renaming approach via structural and lexical analyses». *Frontiers of Information Technology & Electronic Engineering*, 732-748.
- Madsen, O. L. y Møller-Pedersen, B. (2022). «Using Supplementary Properties to Reduce the Need for Documentation». *Springer Nature Switzerland*, 35-59.
- Manrique, J. F. (2007). La Lengua universal de Leibniz. *Saga – Revista de Estudiantes de Filosofía*, 109–119.
- Mansour, M. y Yang, W. (2017). «History of Database Applications». University of Rochester.
- Martraire, C. (2019). *Living Documentation: Continuous Knowledge Sharing*. Addison-Wesley Professional.
- McCandless, G. (2010). *Rhythm and Meter in the Music of Dream Theater. Thesis*.
- McCarthy, J. (1984). «Some Expert System Need Common Sense». *Computer Culture: The Scientific, Intellectual and Social Impact of the Computer*.
- McCauley, R., Grissom, S., Fitzgerald, S. y Murphy, L. (2015). «Teaching and learning recursive programming: a review of the research literature». *Computer Science Education*, 37-66.
- McDermott, J. (1982). «R1: A rule-based configurer of computer systems». *Artificial Intelligence*, 19, 39-88.
- Metz, C. y Collins, K. (marzo de 2023). *10 Ways GPT-4 Is Impressive but Still Flawed* . Obtenido de The New York Times: <https://www.nytimes.com/2023/03/14/technology/openai-new-gpt4.html>
- Microsoft. (2023). *Linux containers on Windows 10*. Obtenido de <https://learn.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/linux-containers>
- Munn, L. (2022). «The uselessness of AI ethics». *AI and Ethics*.

- Naur, P. (1985). «Intuition in software development». *TAPSOFT - Formal Methods and Software Development* (págs. 60-79). Springer Berlin Heidelberg.
- Naur, P. (1985). «Programming as theory building». *Microprocessing and Microprogramming*, 253-261.
- Oliveira, R., Ajala, C., Viana, D., Cafeo, B.y Fontão, A. (2021). «Developer Relations (DevRel) Roles: an Exploratory Study on Practitioners' opinions». *Proceedings of the XXXV Brazilian Symposium on Software Engineering* (págs. 363-367). Association for Computing Machinery.
- Ousterhout, J. K. (2018). *A Philosophy of Software Design*. Yaknyam Press.
- Ozkaya, M.y Erata, F. (2020). «A survey on the practical use of UML for different software architecture viewpoints». *Information and Software Technology*, 106275.
- Perry, N., Srivastava, M., Kumar, D.y Boneh, D. (2022). «Do Users Write More Insecure Code with AI Assistants? ». *arXiv*.
- Polit, S. (1985). «R1 And Beyond: AI Technology Transfer At DEC». *The AI Magazine*, 76-78.
- Pyeatt, L. D.y Ughetta, W. (2020). *ARM 64-Bit Assembly Language*. Newnes.
- Rinderknecht, C. (2014). «A Survey on Teaching and Learning Recursive Programming». *Informatics in Education*.
- Rival, X.y Yi, K. (2020). *Introduction to static analysis an abstract interpretation perspective*. The MIT Press.
- Roy, S., Kot, L.y Koch, C. (2013). «Quantum Databases». *CIDR. Proc. CIDR*.
- Rubio-Sanchez, M. (2017). *Introduction to Recursive Programming*. CRC Press.
- Rubio-Sánchez, M., Urquiza-Fuentes, J.y Pareja-Flores, C. (2008). «A gentle introduction to mutual recursion». *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, (págs. 235-239).

- Russell, S. (2015). «Robotics: Ethics of artificial intelligence». *Nature*.
- Sakharovskiy, K. (2022). «Universal Quantum Gates». *Bachelors Thesis in Information Technology*. University of Jyväskylä.
- Sedgewick, R.y Flajolet, P. (2013). *An Introduction to the Analysis of Algorithms*. Addison-Wesley Professional.
- Shariffdeen, R., Noller, Y., Grunske, L.y Roychoudhury, A. (2021). «Concolic Program Repair». *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (págs. 390-405). Association for Computing Machinery.
- Shore, J. a. (2007). *The Art of Agile Development*. O'Reilly.
- Solomonoff, G. (2013). «Ray Solomonoff and the New Probability». En *Lecture Notes in Computer Science* (págs. 37–52).
- Spinellis, D. (2012). *Coding Standards and Conventions*. Obtenido de <https://www.spinellis.gr/ismr/conv/indexw.htm>
- Stallman, R. (2001). *GNU coding standards*. Obtenido de [http://www.gnu.org/prep/standards\\_toc.html](http://www.gnu.org/prep/standards_toc.html)
- Stroustrup, B. (1995). «Why C++ is not just an object-oriented programming language». *OOPS Messenger*.
- Sulov, V. (2016). «Iteration vs recursion in introduction to programming classes: an empirical study». *Cybern. Inf. Technol.*, 63-72.
- Sutter, H. a. (2004). *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices*. Addison-Wesley Professional.
- Talib, M. A. (2021). «A systematic literature review on hardware implementation of artificial intelligence algorithms». *The Journal of Supercomputing*, 1897-1938.
- Tanenbaum, A. S.y Bos, H. (2014). *Modern Operating Systems*. Pearson.

The Guardian. (2020). *A robot wrote this entire article. Are you scared yet, human?*

Obtenido

de

<https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3>

The Linux Foundation. (2015). *10 Years of Git: An Interview with Git Creator Linus Torvalds*. Obtenido de THE LINUX FOUNDATION:

<https://www.linuxfoundation.org/blog/blog/10-years-of-git-an-interview-with-git-creator-linus-torvalds>

Thengvall, M. (2018). *The Business Value of Developer Relations*. Apress.

Thompson, K. (1984). «Reflections on Trusting Trust». *ACM Turing lecture*.

Thrun, S. (2007). «Stanley: The Robot That Won the DARPA Grand Challenge». *Springer Berlin Heidelberg*.

Toffoli, T. (2005). «Reversible computing». *Springer, Berlin, Heidelberg*.

Turner, R. (2018). *Computational Artifacts: Towards a Philosophy of Computer Science*. Springer.

Vailshery, L. S. (2022). *Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030*. Obtenido de statista: <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>

van Steen, M.y Tanenbaum, A. S. (2016). «A brief introduction to distributed systems». *Computing*, 967–1009.

Vleck, T. V. (1995). *The IBM 7094 and CTSS*. Obtenido de <https://multicians.org/thvv/7094.html>

Vo, T., Dave, P., Bajpai, G.y Kashef, R. (2022). «Edge, Fog, and Cloud Computing : An Overview on Challenges and Applications». *arXiv*.

- Wake, W. C. (2003). *Refactoring Workbook*. Addison-Wesley Longman Publishing Co., Inc.
- Weizenbaum, J. (1976). *Computer Power and Human Reason: From Judgment to Calculation*. W. H. Freeman and Company.
- Wing, J. M. (2006). «Computational Thinking». *Communications of the ACM*, 33--35.
- Wolfram, MathWorld. (s.f.). *Elementary Cellular Automaton* . Obtenido de <https://mathworld.wolfram.com/ElementaryCellularAutomaton.html>
- Wolfram, S. (1984). «Computation theory of cellular automata». *Commun. Math. Phys.*
- Wooldridge, M. (2021). *A Brief History of Artificial Intelligence: What It Is, Where We Are, and Where We Are Going*. Flatiron Books.
- Zhi, J. a. (2014). «Cost, Benefits and Quality of Software Development Documentation: A Systematic Mapping». *Journal of Systems and Software*.
- Zickert, F. (2021). *Hands-On Quantum Machine Learning With Python. Volume 1: Get Started*.



# ÍNDICE DE NOMBRES

## A

Abelson, Harold, 516  
Adzic, Gojko, 306  
Aghajani, Emad, 298  
Aho, Alfred, 360, 365, 491, 517  
Altman, Sam, 440  
Amdahl, Gene, 257  
Aristóteles, 328, 503  
Arpaci-Dusseau, Andrea C., 377, 388  
Arpaci-Dusseau, Remzi H., 377, 388

## B

Babbage, Charles, 130, 345  
Bach, Johann Sebastian, 526  
Bachman, Charles, 391  
Backus, John, 346  
Bacon, Francis, 155  
Bailis, Peter, 404  
Bengio, Yoshua, 433, 436, 501  
Bentley, Jon, 313, 517  
Berners-Lee, Tim, 406  
Bezanson, Jeff, 349  
Biancuzzi, Federico, 518  
Blair, Gordon, 424  
Booch, Grady, 517  
Boole, George, 130, 137  
Boyce, Raymond F., 392  
Brooks, Frederick, 465  
Bruderer, Herbert, 345  
Budgen, David, 159  
Burkov, Andriy, 501

## C

Chacon, Scott, 485  
Chamberlin, Donald D., 392  
Chen, Peter, 392  
Codd, Edgar, 391, 397, 499  
Cohen, Bram, 407  
Conway, John Horton, 226  
Corbató, Fernando, 369  
Cormen, Thomas H., 516  
Cortes, Corinna, 435  
Coulouris, George, 423, 424  
Courville, Aaron, 501

## D

Daley, Bob, 369  
Darwin, Charles, 519  
Date, C. J., 397, 404, 499  
Dean, Jeffrey, 260  
Denning, Peter J., 129, 136, 139, 471  
Dijkstra, Edsger, 128, 136, 347, 461, 493, 530  
Dobelli, Rolf, 525  
Dollimore, Jean, 424  
Dream Theater, 526

## E

Epicteto, 533  
Escher, Maurits Cornelis, 527

## F

Feigenbaum, Edward, 431  
Fenner, Bill, 517  
Feynman, Richard, 501  
Flower, Martin, 172  
Fogel, David B., 512  
Fukushima, Kunihiro, 436

## G

Gamma, Erich, 517  
Ghemawat, Sanjay, 260  
Gödel, Kurt, 137, 461, 526  
Gogh, Vincent van, 155  
Goldreich, Oded, 232  
Goodfellow, Ian, 501  
Gosling, James, 348

## H

Han, Byung-Chul, 531  
Hejlsberg, Anders, 349  
Hellerstein, Joseph M., 404  
Helm, Richard, 517  
Hennessy, John L., 516  
Hilbert, David, 462  
Hintjens, Pieter, 504, 510  
Hinton, Geoffrey, 433, 437  
Ho, Tin Kam, 435  
Hoare, Antony, 128, 294  
Hoare, Graydon, 349  
Hofstadter, Douglas, 526  
Hopfield, John, 432  
Hopper, Grace, 346  
Høst, Einar W., 303  
Hunt, Andy, 477  
Huxley, Aldous, 495

## J

Jacquard, Joseph-Marie, 345  
Jerzyk, Marcel, 177  
Johnson, Ralph, 517

## K

Kasparov, Gary, 435, 437  
Kay, Alan, 348, 356  
Kemeny, John, 347  
Kempelen, Wolfgang von, 345  
Kernighan, Brian, 360  
Kerrisk, Michael, 497  
Kindberg, Tim, 424  
Knuth, Donald, 109, 188, 516  
Kurtz, Thomas, 347

## L

Lampert, Leslie, 461, 493, 504, 510, 522  
LeCun, Yann, 433, 436  
Lederberg, Joshua, 431  
Leibniz, 129, 130, 345  
Leiserson, Charles E., 516  
Lighthill, James, 432  
Lovelace, Ada, 344, 345

## M

Madsen, Ole Lehrmann, 299  
Mahon, Charles, 345  
McCarthy, John, 346, 428, 434  
McConnell, Steve, 517  
Merwin-Daggett, Marjorie, 369  
Michalewicz, Zbigniew, 512  
Minsky, Marvin, 428, 431, 432  
Møller-Pedersen, Birger, 299  
Morgenstern, Oskar, 427  
Morland, Sir Samuel, 345  
Murphy, Kevin P., 501

Musk, Elon, 440

## N

Naur, Peter, 188, 346

Navathe, Shamkant B., 517

Newell, Allen, 428

Norvig, Peter, 447, 500, 516

Nystrom, Robert, 365

## Ø

Østvold, Bjarte M., 303

## O

Ousterhout, John, 159, 195, 467

## P

Papert, Seymour, 431, 432

Pascal, Blaise, 345

Patterson, David A., 516

Patterson, Richard North, 494

Peano, Giuseppe, 137

Pearson, Matt, 529

Peirce, Charles Sanders, 137

Petzold, Charles, 517

Pike, Rob, 349

Pin, Víctor Gómez, 533

Pla, Josep, 455

Platón, 328, 457

Pólya, George, 512

Pressman, Roger S., 462

Prometeo, 327

Pyeatt, Larry, 204

## R

Ramez Elmasri, 517

Ramon y Cajal, Santiago, 519

Ravel, Maurice, 167

Ritchie, Dennis, 333, 347, 370

Rivest, Ronald, 516

Rosenblatt, Frank, 428, 432

Rossum, Guido van, 348

Rudoff, Andrew M., 517

Russell, Bertrand, 428, 455, 519, 523

Russell, Stuart, 329, 447, 500, 516

## S

Samuel, Arthur, 427, 428, 429

Seibel, Peter, 518

Séneca, 455, 533

Sethi, Ravi, 517

Shannon, Claude, 130, 428

Shiffman, Daniel, 228, 498

Silberschatz, Abraham, 497, 517

Simon, Herbert, 428

Sipser, Michael, 516

Sócrates, 328, 330

Solomonoff, Ray, 428, 429

Sommerville, Ian, 462

Spinellis, Diomidis, 482

Stallman, Richard, 482

Stein, Clifford, 516

Stevens, William Richard, 517

Stonebraker, Michael, 393, 404

Strachey, Christopher, 427

Straub, Ben, 485

Strauch, Christof, 499

Stroustrup, Bjarne, 348

Sussman, Gerald Jay, 516

Sussman, Julie, 516

## T

Tanenbaum, Andrew, 372, 388, 416, 424,  
488

Tedre, Matti, 129, 136, 139, 471

Thomas, Dave, 477

Thompson, Ken, 333, 347, 349, 370

## Principios de programación

Torvalds, Linus, 372, 484, 488  
Tucídides, 492  
Turing, Alan, 426, 455  
Turner, Raymond, 341

## U

Ughetta, William, 204  
Ullman, Jeffrey, 365, 517

## V

Van Steen, Maarten, 416, 424  
Vapnik, Vladimir, 435  
Vlissides, John, 517  
Von Neumann, John, 427

## W

Wake, William, 176  
Warden, Shane, 518  
Weizenbaum, Joseph, 364, 430  
Whitehead, Alfred North, 428  
Williams, Ronald J., 433  
Wing, Jeannette M., 128  
Wirth, Niklaus, 347  
Wittgenstein, Ludwig, 44, 464, 523  
Wolfram, Stephen, 223  
Wooldridge, Michael, 438

## Z

Zeus, 327

# ÍNDICE TEMÁTICO

## A

Aarch64, 203, 206, 207, 208, 209, 211, 213, 216  
ACM Library, 524  
Agda, 322, 350  
Algol, 96, 131, 137, 346  
algoritmo voraz, 116  
algoritmos seriales, 257  
Alhambra, 528  
AlphaFold, 440, 445  
AlphaGo, 438  
AlphaZero, 438  
análisis estático, 310, 316, 318, 319, 320, 321, 322, 326  
Android, 374, 375, 376, 384  
API Rest, 40, 489  
Apple Lisa, 371  
aprendizaje automático, 178, 241, 304, 310, 323, 348, 357, 359, 394, 395, 402, 403, 426, 427, 429, 430, 431, 433, 434, 435, 436, 437, 438, 441, 442, 443, 444, 447, 501  
aprendizaje profundo, 125, 192, 242, 259, 357, 426, 437, 439, 440, 444, 445, 446, 501  
ARM, 202, 203, 204, 205, 206, 227  
arXiv, 524  
autómata celular, 223, 224, 225, 226, 228  
autómata finito *determinista*, 218  
autómata finito no-determinista, 222  
AWK, 343, 360, 480, 485, 486, 487  
AWS, 522

## B

B5000, 138  
B6700, 138  
Bash, 479  
BASIC, 347  
Big Data, 394  
Bing Chat, 328, 445  
BitTorrent, 407  
*Blockchain*, 125, 314, 422, 423, 468  
Boston Dynamic, 446  
BPMN, 191  
bucles anidados, 85

## C

C, 347  
C#, 145, 168, 172, 173, 178, 179, 208, 349, 356, 361, 494, 498  
C++, 39, 57, 82, 119, 145, 172, 201, 204, 208, 255, 262, 269, 318, 343, 348, 350, 353, 356, 357, 364, 378, 469, 470, 481, 488, 497, 498  
CentOS, 373  
ChatGPT, 328, 414, 425, 441, 445  
ChuckK, 527  
*code smells*, 176, 177  
complejidad cuadrática, 236  
complejidad exponencial, 237  
complejidad lineal, 233, 235, 236, 240  
complejidad logarítmica, 239  
computación de frontera, 408  
computación en la niebla, 408

computación frontera, 242  
computación reversible, 286  
criptografía, 41, 225, 314, 407, 423

## D

DALL·E, 425  
data race, 145, 383  
*deadlock*, 69, 254, 321, 398, 399  
DeepMind, 438, 439  
dividir y conquistar, 113  
Docker, 375, 376, 384  
DOT, 353  
DRAM, 206, 384

## E

ELIZA, 430  
evaluación perezosa, 354  
exclusión mutua, 251, 421

## F

Filosofía del software, 456  
FLOW-MATIC, 346  
FORTRAN, 346, 350, 352, 356, 368, 434  
FPGA, 242  
Friden Flexowriter, 369  
fuerza bruta, 110, 112, 231  
funciones de orden superior, 354

## G

git, 483  
GitHub Copilot, 193, 364, 445, 489, 490  
GNOME, 376, 480  
GO TO, 136  
Golang, 349  
Google Brain, 439  
GPT-3, 440

## H

Haskell, 322, 350, 357, 459, 468, 470, 495  
HPC, 314  
htop, 480  
HTTP, 251, 261, 406  
Hyper-V, 376

## I

IBM 650, 368  
IBM Watson, 438  
Intel x86, 203, 205  
iOS, 374, 384, 437  
iTunes, 374

## J

Java, 57, 70, 145, 207, 208, 255, 303, 343,  
348, 356, 357, 364, 398, 460, 469, 499  
JavaScript, 255, 269, 281, 289, 319, 349,  
353, 354, 470, 494, 497, 498, 515  
jQuery, 354  
Juego de la Vida, 226  
Julia, 56, 70, 119, 349, 353, 500

## L

LaTeX, 188, 342, 343, 353  
lenguajes de programación probabilísticos,  
358, 364  
LevelDB, 488  
ley de Amdahl, 257  
Linux, 334, 372, 373, 375, 376, 480, 488,  
497  
Lisp, 131, 133, 137, 346, 442, 468  
lista enlazada, 122  
llamada a procedimiento remoto, 262  
LSTM, 439

**M**

MapReduce, 260  
 máquina de Turing, 105, 136, 469  
 Markdown, 342  
 Matlab, 500  
 MBSE, 192  
 Melröse, 527  
 memoización, 114, 115  
 metaprogramación, 145, 348  
 metodología agile, 298  
 microservicios, 413  
 MINIX, 372, 488  
 MIPS, 202, 205  
 Modelos de computación, 198  
 MongoDB, 401  
 MPI, 261  
 multiprocesamiento simétrico, 371

**N**

Napster, 407  
 Naturalize, 302, 303  
 Neuro-Symbolic, 445  
 NewSQL, 390, 394  
 NoSQL, 340, 390, 393, 394, 400, 401, 402, 404, 499  
 NP-Hard, 231  
 Numpy, 122

**O**

ODBMS, 392  
 OpenAI, 440, 441, 447, 490  
 openSUSE, 373

**P**

P2P, 407  
 Pascal, 347  
 patrón de diseño, 178, 179, 182, 184, 194

PDP-11, 205, 370  
 Pensamiento Computacional, 127, 128, 129, 131, 139  
 Perl, 480, 485  
 pila, 98, 101, 120, 133, 138, 207, 208, 212, 215, 320  
 PlantUML, 353  
 Postgres, 393  
 PowerPC, 205  
 premio Turing, 188, 333, 360, 428, 431, 433, 491, 493, 504, 530  
 Processing, 205, 242, 260, 497, 498, 529  
 profilers, 244  
 programación de bajo nivel, 198  
 programación dinámica, 114  
 programación genérica, 170  
 Prolog, 434  
 protocolos de comunicación, 411, 504  
 pruebas dinámicas, 310, 311, 325, 461  
 pruebas estáticas, 310, 316, 326  
 PyTorch, 515

**Q**

Quora, 503

**R**

RabbitMQ, 418, 419, 420, 421  
 Racket, 347, 495  
 React, 343, 354  
 realidad aumentada, 138, 386  
 realidad virtual, 386  
 recursión anidada, 103  
 recursión de cola, 98  
 recursión lineal, 97  
 recursión múltiple, 100  
 recursión mutua, 102  
 Regex, 342  
 ResearchGate, 524  
 RISC, 202, 205, 227, 247

RISC-V, 205, 227, 247  
RNN, 439  
robots, 330  
RocksDB, 488  
RPC, 263, 417, 418  
Ruby and Rails, 354  
Rust, 57, 119, 120, 121, 145, 255, 349, 353,  
468, 481, 497

## S

Scala, 499  
Simula 67, 133  
sistemas operativos, 28, 36, 138, 309, 497  
Smalltalk, 348, 356, 392  
SOLID, 460  
Sonic Pi, 527  
Springer, 524  
SRAM, 206  
Supercomputación, 242  
SyBase, 393  
SysML, 191  
System R, 392

## T

TCP/IP, 261, 406, 411  
Teselación de Penrose, 528  
TLA+, 342, 522  
tmux, 480  
TOGAF, 192  
Transformer, 439  
transparencia referencial, 354, 355, 470  
Turing completo, 342

Twitch, 508  
Twitter, 197, 503, 505, 508, 514  
TypeScript, 319, 349

## U

Ubuntu, 373, 480  
UML, 159, 190, 191, 353  
UNIVAC I, 368  
UNIX, 57, 333, 339, 347, 369, 370, 372,  
384, 387, 411, 413, 479, 482, 487, 488,  
497

## V

Vim, 480  
virtualización, 242, 368, 371, 375, 377,  
378, 379, 381, 382, 384, 407  
Visual Studio Code, 334, 349, 441, 479  
vuelta atrás, 117

## W

WebSocket, 261  
Windows 7, 374  
Windows Vista, 374  
Windows XP, 373, 374

## Y

YouTube, 508

## Z

ZeroMQ, 262, 308, 505