



# ESTRUCTURA DE LOS LENGUAJES

Introducción

Dr. Christian von Lücken



# TEMAS DEL CURSO

- ▶ Introducción a los Lenguajes de Programación
- ▶ Nombres, Vinculaciones, Alcance
- ▶ Sintaxis y Semántica
- ▶ Lex y Yacc
- ▶ Lenguajes Funcionales
- ▶ Tipos de datos
- ▶ Expresiones, asignaciones, declaraciones de control
- ▶ Subprogramas
- ▶ Ejecución de Subprogramas
- ▶ Lenguajes Lógicos
- ▶ Lenguajes funcionales



# ESTRUCTURA DE LOS LENGUAJES

Razones para estudiar estructura de los lenguajes de programación

Dr. Christian von Lücken

# OBJETIVOS

1. Analizar porqué son necesarios los lenguajes de alto nivel
2. Identificar los lenguajes de programación más utilizados
3. Analizar los motivos por los cuáles estudiamos lenguajes de programación
4. Analizar las clasificaciones de los lenguajes de programación

# MIPS R4000

## LENGUAJE MÁQUINA



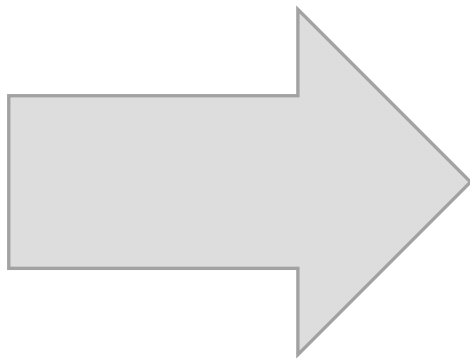
```
27bdf fd0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483fffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```

Tarea: Describir el proceso computacional  
llevado a cabo

# MIPS R4000

## LENGUAJE MÁQUINA VS ENSAMBLADOR

```
27bdfdd0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483fffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```



```
addiu    sp,sp,-32
sw       ra,20(sp)
jal      getint
nop
jal      getint
sw       v0,28(sp)
lw       a0,28(sp)
move     v1,v0
beq      a0,v0,D
slt      at,v1,a0
A: beq    at,zero,B
nop
```

```
        b      C
        subu    a0,a0,v1
B:      subu    v1,v1,a0
C:      bne     a0,v1,A
        slt     at,v1,a0
D:      jal     putint
        nop
        lw      ra,20(sp)
        addiu   sp,sp,32
        jr      ra
        move    v0,zero
```

# ¿QUÉ ES UN LENGUAJE DE PROGRAMACIÓN?

Una notación formal y rigurosa para describir **computaciones** de una forma legible tanto para la máquina como para el ser humano.

Computación cualquier proceso que pueda ser efectuado por una computadora, incluye todo tipo de procesamiento en una computadora.

# ¿QUÉ ES PROGRAMAR?

La programación es el **arte** de decirle a otro humano que es lo que se desea que haga una computadora

Donald Knuth





# PROGRAMACIÓN Y LENGUAJES DE PROGRAMACIÓN

El arte de la programación es el **arte de organizar la complejidad**, ... debemos organizar los cálculos de manera que nuestros limitados sentidos sean suficientes para garantizar que el cómputo arroje los resultados esperados.

El **lenguaje** debe ayudarnos a escribir **buenos programas**; un programa es bueno si es **fácil de leer, fácil de entender y fácil de modificar**

Ravi Sethi, Lenguajes de Programación



# ¿PORQUÉ ESTUDIAR LENGUAJES DE PROGRAMACIÓN?



# ¿PORQUÉ ESTUDIAR LENGUAJES DE PROGRAMACIÓN?

- Uno o dos lenguajes no son suficientes para un profesional de la informática
- Usted debe saber
  - los conceptos generales subyacentes a las características de los lenguajes
  - opciones en el diseño de lenguajes de programación.





# RAZONES PARA ESTUDIAR LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

1. **Incrementar la capacidad de expresar ideas**
2. Mejorar la capacidad de elegir lenguajes apropiados
3. Incrementar la habilidad para aprender nuevos lenguajes
4. Mejorar el entendimiento del significado de la implementación
5. Mejorar el uso de lenguajes que son conocidos
6. Avance general de la computación

# INCREMENTAR LA CAPACIDAD PARA EXPRESAR IDEAS

Lenguajes naturales:

- ▶ La profundidad a la que la gente piensa está influenciada por el poder expresivo de la lengua.
- ▶ Cuantas más construcciones apropiadas tengas, más fácil será comunicar.
- ▶ Es difícil para las personas conceptualizar estructuras que no pueden describir verbalmente.

# INCREMENTAR LA CAPACIDAD PARA EXPRESAR IDEAS

Lenguajes de programación:

- ▶ Esto es similar para los PL. El lenguaje en el que desarrolla el software pone límites a los tipos de estructuras de datos, estructuras de control y abstracciones que se pueden utilizar.
- ▶ El conocimiento de una variedad más amplia de características del lenguaje de programación puede reducir tales limitaciones en el desarrollo de software.
- ▶ Los programadores aumentan su capacidad para el desarrollo de software mediante el aprendizaje de nuevas construcciones de lenguaje. Por ejemplo, si aprende matrices asociadas en Perl, puede simularlas en C.

# RAZONES PARA ESTUDIAR LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

- 1. Incrementar la capacidad de expresar ideas**
- 2. Mejorar la capacidad de elegir lenguajes apropiados**
3. Incrementar la habilidad para aprender nuevos lenguajes
4. Mejorar el entendimiento del significado de la implementación
5. Mejorar el uso de lenguajes que son conocidos
6. Avance general de la computación



# MEJORAR EL CONOCIMIENTO PARA ELEGIR LENGUAJES APROPIADOS

- ▶ No todos los lenguajes de programación pueden ser adecuados para una implementación. Los programadores familiarizados con una amplia gama de lenguajes están más preparados para elegir el más adecuado.
- ❑ Muchos programadores aprenden uno o dos lenguajes específicos de los proyectos.
- ❑ Es posible que ya no se utilicen algunos de esos.
- ❑ Cuando inician un nuevo proyecto continúan utilizando aquellos lenguajes que son antiguos y no se adaptan a los proyectos actuales.

# MEJORAR EL CONOCIMIENTO PARA ELEGIR LENGUAJES APROPIADOS

- ❑ Sin embargo, otro lenguaje puede ser más apropiado para la naturaleza del proyecto.
  - Mucho procesamiento de texto -> Perl puede ser una buena opción.
  - Gran cantidad de procesamiento matricial -> Se puede utilizar MATLAB.
- ❑ Estudiar los principios de los lenguajes de programación proporciona una manera de juzgarlos:
  - “Las ventajas de Perl para este problema son...”,
  - “Las ventajas de Java son...”

# RAZONES PARA ESTUDIAR LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

- 1. Incrementar la capacidad de expresar ideas**
- 2. Mejorar la capacidad de elegir lenguajes apropiados**
- 3. Incrementar la habilidad para aprender nuevos lenguajes**
4. Mejorar el entendimiento del significado de la implementación
5. Mejorar el uso de lenguajes que son conocidos
6. Avance general de la computación

# INCREMENTAR LA HABILIDAD PARA APRENDER NUEVOS LENGUAJES

- ▶ Los lenguajes de programación siguen evolucionando
  - Muchos son muy nuevos, se crearan nuevos lenguajes con el tiempo.
- ▶ Si conoce los conceptos de los lenguajes de programación, aprenderá otros lenguajes mucho más fácilmente.
  - Por ejemplo, si conoce el concepto de programación orientada a objetos, es más fácil aprender C++ después de aprender Java

Al igual que los lenguajes naturales,

- Cuanto mejor conozcas la gramática de un idioma, más fácil te resultará aprender un segundo idioma.
- Aprender nuevos idiomas permite que aprendas cosas sobre los idiomas que ya conoces

# LENGUAJES MAS UTILIZADOS JUNIO 2020





















[HTTPS://WWW.TIOBE.COM/TIOBE-INDEX/](https://www.tiobe.com/tiobe-index/)

Jun 2020	Jun 2019	Programming Language	Ratings	Change
1	2	C	17.19%	+3.89%
2	1	Java	16.10%	+1.10%
3	3	Python	8.36%	-0.16%
4	4	C++	5.95%	-1.43%
5	6	C#	4.73%	+0.24%
6	5	Visual Basic	4.69%	+0.07%
7	7	JavaScript	2.27%	-0.44%
8	8	PHP	2.26%	-0.30%
9	22	R	2.19%	+1.27%
10	9	SQL	1.73%	-0.50%
11	11	Swift	1.46%	+0.04%
12	15	Go	1.02%	-0.24%
13	13	Ruby	0.98%	-0.41%
14	10	Assembly language	0.97%	-0.51%
15	18	MATLAB	0.90%	-0.18%
16	16	Perl	0.82%	-0.36%
17	20	PL/SQL	0.74%	-0.19%
18	26	Scratch	0.73%	+0.20%
19	19	Classic Visual Basic	0.65%	-0.42%
20	38	Rust	0.64%	+0.38%



# LENGUAJES MAS UTILIZADOS JUNIO 2022



















[HTTPS://WWW.TIOBE.COM/TIOBE-INDEX/](https://www.tiobe.com/tiobe-index/)

Jul 2022	Jul 2021	Change	Programming Language		Ratings	Change
1	3	▲		Python	13.44%	+2.48%
2	1	▼		C	13.13%	+1.50%
3	2	▼		Java	11.59%	+0.40%
4	4			C++	10.00%	+1.98%
5	5			C#	5.65%	+0.82%
<b>6</b>	<b>6</b>			<b>Visual Basic</b>	<b>4.97%</b>	<b>+0.47%</b>
7	7			JavaScript	1.78%	-0.93%
8	9	▲		Assembly language	1.65%	-0.76%
9	10	▲		SQL	1.64%	+0.11%
10	16	▲		Swift	1.27%	+0.20%
11	8	▼		PHP	1.20%	-1.38%
<b>12</b>	<b>13</b>	▲		<b>Go</b>	<b>1.14%</b>	<b>-0.03%</b>
13	11	▼		Classic Visual Basic	1.07%	-0.32%
14	20	▲		Delphi/Object Pascal	1.06%	+0.21%
15	17	▲		Ruby	0.99%	+0.04%
16	21	▲		Objective-C	0.94%	+0.17%
17	18	▲		Perl	0.78%	-0.12%
18	14	▼		Fortran	0.76%	-0.36%
19	12	▼		R	0.76%	-0.57%
20	19	▼		MATLAB	0.73%	-0.15%



# LENGUAJES MAS UTILIZADOS FEBRERO 2023

[HTTPS://WWW.TIOBE.COM/TIOBE-INDEX/](https://www.tiobe.com/tiobe-index/)

Feb 2023	Feb 2022	Change	Programming Language		Ratings	Change
1	1			Python	15.49%	+0.16%
2	2			C	15.39%	+1.31%
3	4	^		C++	13.94%	+5.93%
4	3	v		Java	13.21%	+1.07%
5	5			C#	6.38%	+1.01%
6	6			Visual Basic	4.14%	-1.09%
7	7			JavaScript	2.52%	+0.70%
8	10	^		SQL	2.12%	+0.58%
9	9			Assembly language	1.38%	-0.21%
10	8	v		PHP	1.29%	-0.49%
11	11			Go	1.11%	-0.12%
12	13	^		R	1.08%	-0.04%
13	14	^		MATLAB	0.99%	-0.04%
14	15	^		Delphi/Object Pascal	0.95%	+0.05%
15	12	v		Swift	0.93%	-0.25%
<b>16</b>	<b>16</b>			<b>Ruby</b>	<b>0.83%</b>	<b>-0.06%</b>
17	19	^		Perl	0.79%	-0.01%
18	22	^^		Scratch	0.76%	+0.13%
19	17	v		Classic Visual Basic	0.74%	-0.09%
20	24	^^		Rust	0.70%	+0.16%

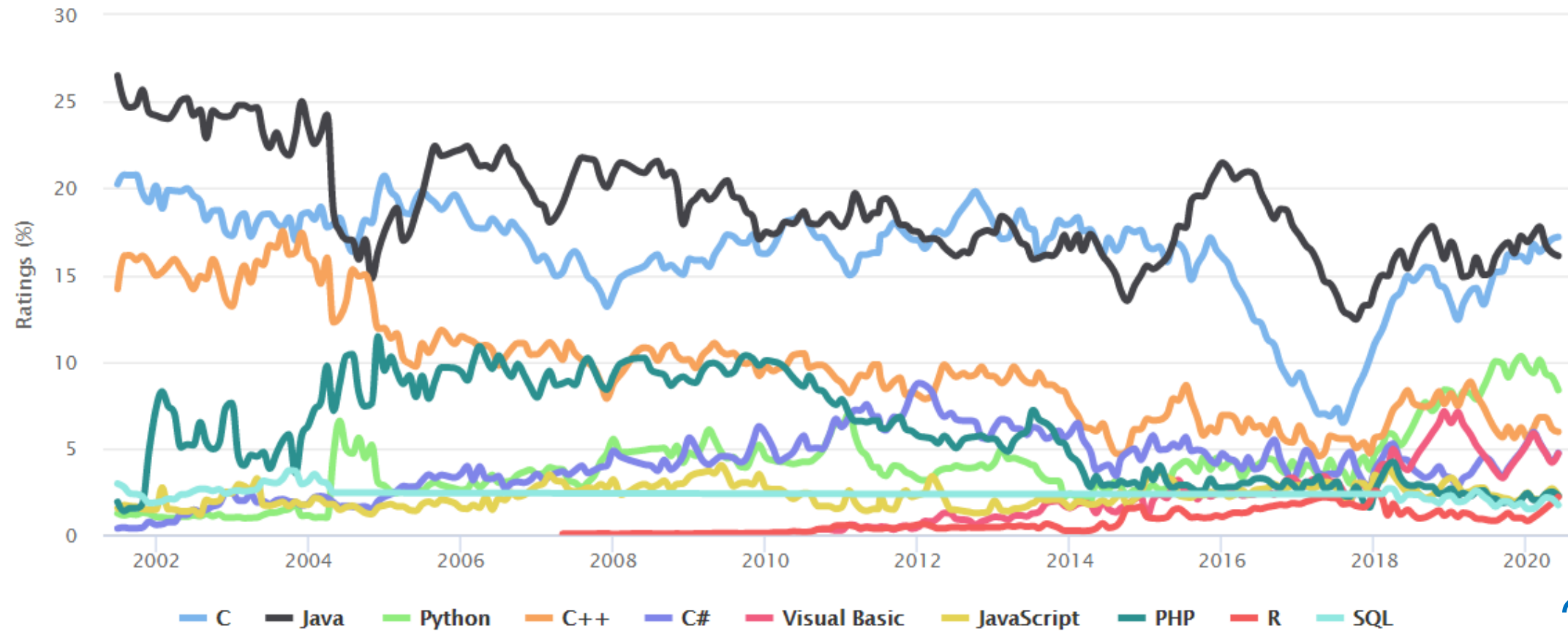


# LENGUAJES MÁS UTILIZADOS JUNIO 2020



## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



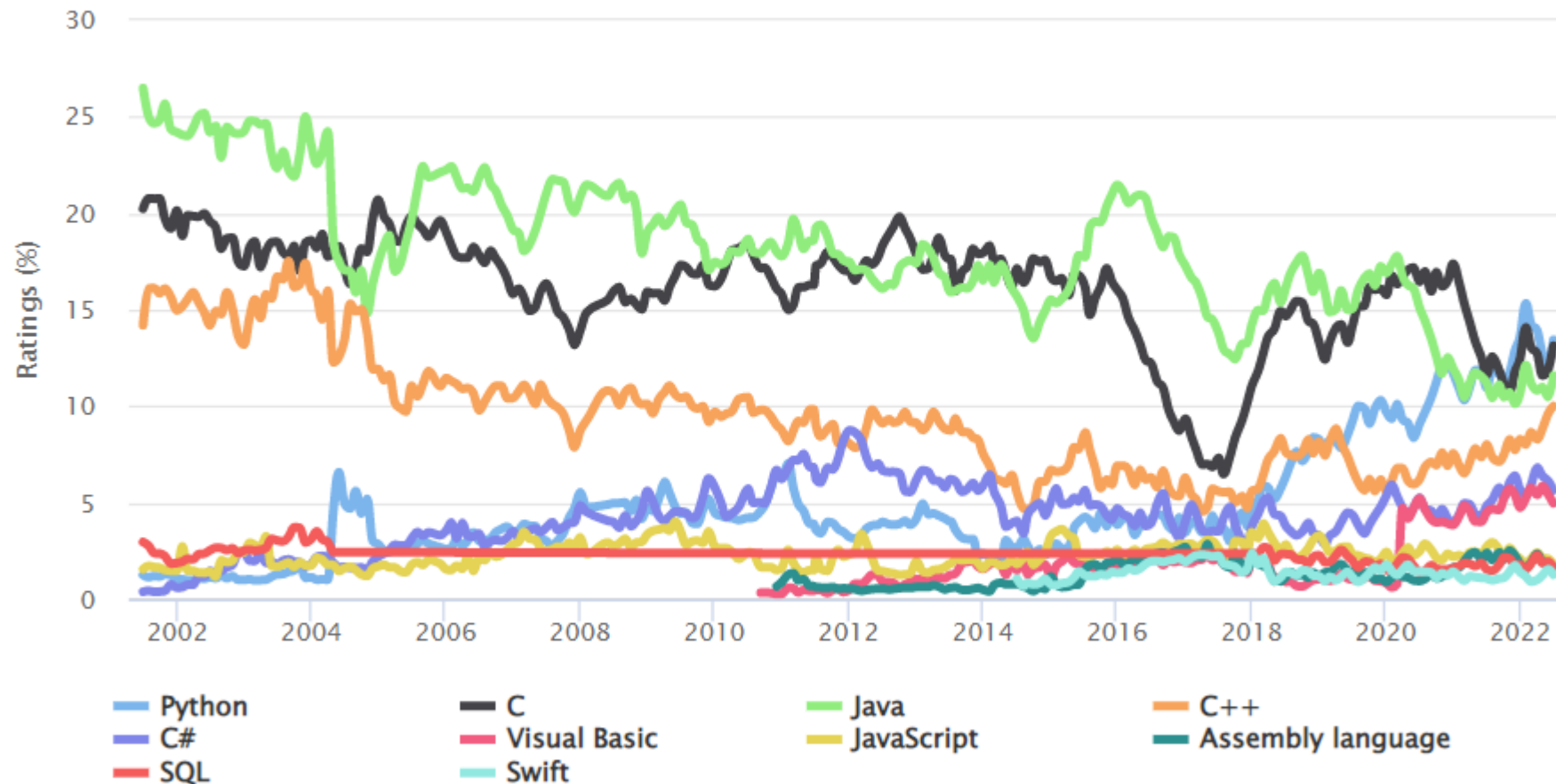


# LENGUAJES MÁS UTILIZADOS JUNIO 2022

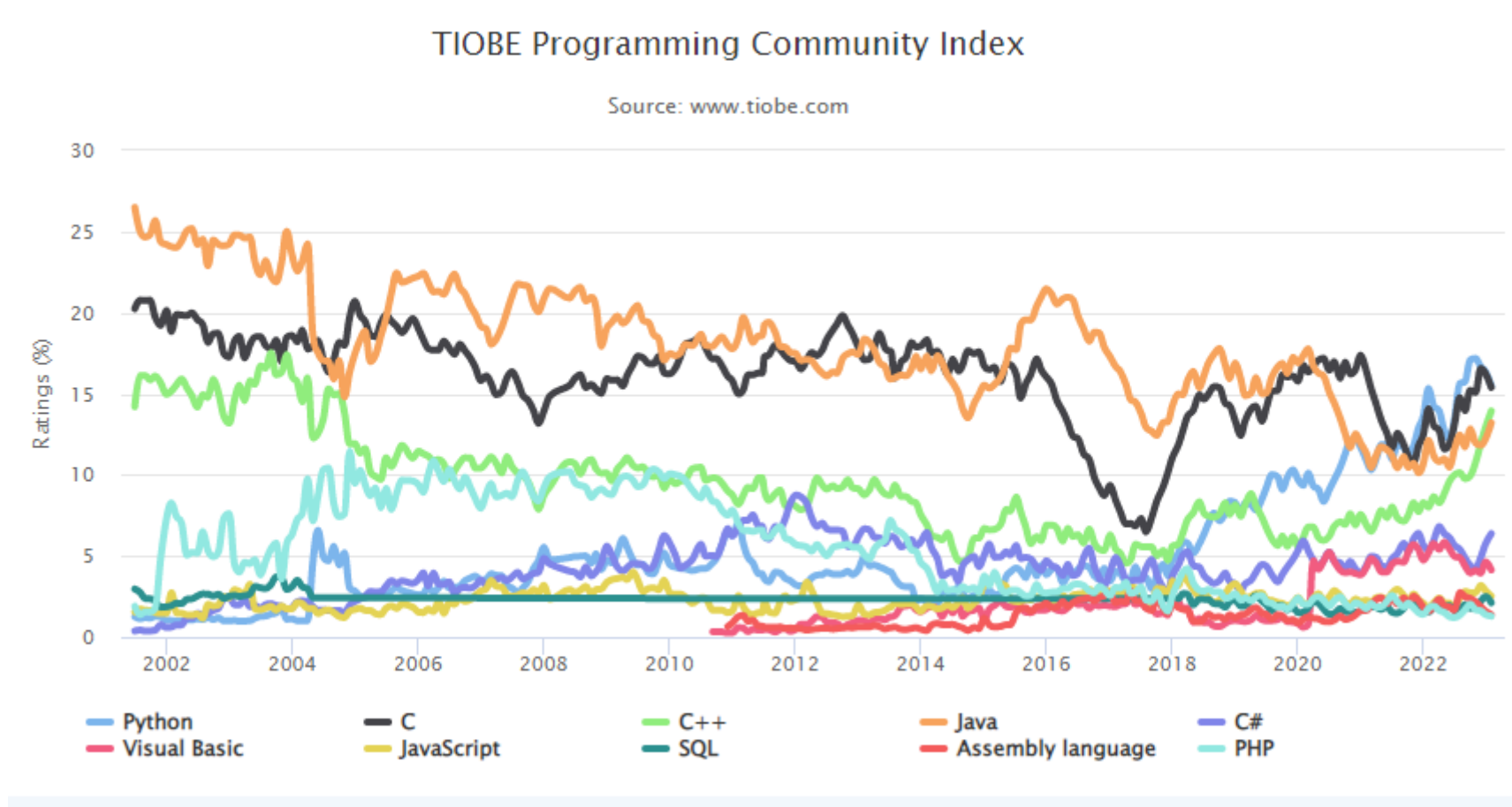


## TIOBE Programming Community Index











Source: [www.tiobe.com](http://www.tiobe.com)








# LENGUAJES MÁS UTILIZADOS FEBRERO 2023



# LENGUAJES MÁS UTILIZADOS FEBRERO 2024

Feb 2024	Feb 2023	Change	Programming Language		Ratings	Change
1	1			Python	15.16%	-0.32%
2	2			C	10.97%	-4.41%
3	3			C++	10.53%	-3.40%
4	4			Java	8.88%	-4.33%
5	5			C#	7.53%	+1.15%
6	7	^		JavaScript	3.17%	+0.64%
7	8	^		SQL	1.82%	-0.30%
8	11	^		Go	1.73%	+0.61%
9	6	v		Visual Basic	1.52%	-2.62%
10	10			PHP	1.51%	+0.21%

# LENGUAJES MÁS UTILIZADOS AGOSTO 2024

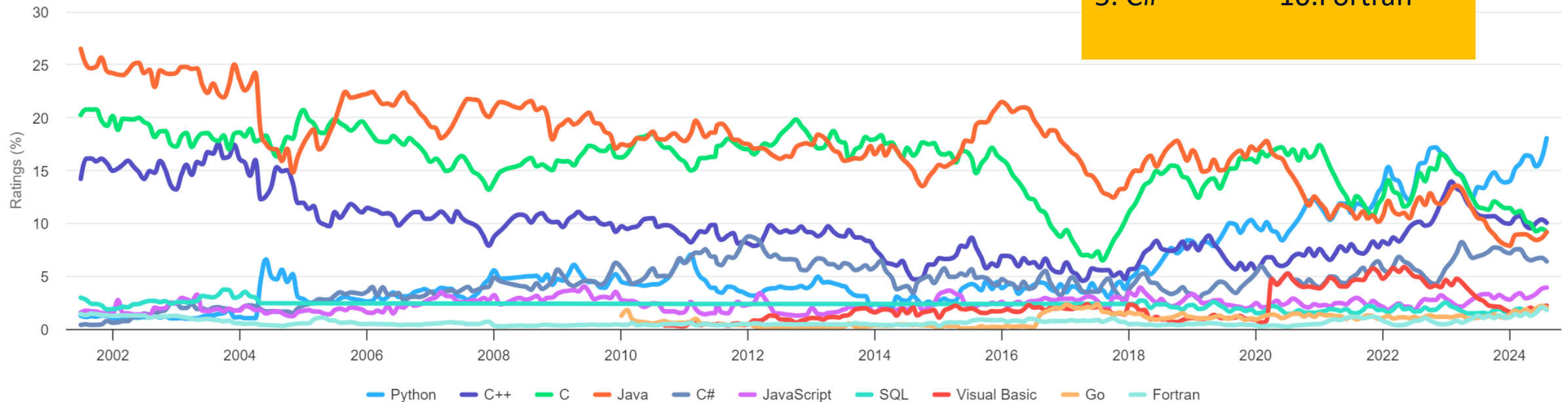
Aug 2024	Aug 2023	Change	Programming Language		Ratings	Change
1	1			Python	18.04%	+4.71%
2	3	⬆		C++	10.04%	-0.59%
3	2	⬇		C	9.17%	-2.24%
4	4			Java	9.16%	-1.16%
5	5			C#	6.39%	-0.65%
6	6			JavaScript	3.91%	+0.62%
7	8	⬆		SQL	2.21%	+0.68%
8	7	⬇		Visual Basic	2.18%	-0.45%
9	12	⬆		Go	2.03%	+0.87%
10	14	⬆		Fortran	1.79%	+0.75%
11	13	⬆		MATLAB	1.72%	+0.67%
12	23	⬆		Delphi/Object Pascal	1.63%	+0.83%
13	10	⬇		PHP	1.46%	+0.19%

# LENGUAJES MÁS UTILIZADOS FEBRERO 2024

- |           |                 |
|-----------|-----------------|
| 1. Python | 6. JavaScript   |
| 2. C++    | 7. SQL          |
| 3. C      | 8. Visual Basic |
| 4. Java   | 9. Go           |
| 5. C#     | 10. Fortran     |

## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# Why is Fortran Programming Language Popular Again?



Sunil Jain · Follow

6 min read · May 31, 2024

## Key Milestones

Over the decades, Fortran's journey has been marked by significant milestones:

- **Fortran II (1958):** Introduced the concept of subroutines and functions, making it easier to organize, and reuse code.
- **Fortran IV (1962):** Added advanced features like IF statements and data types, further enhancing its capability.
- **Fortran 77 (1978):** Standardized the language, ensuring consistency across different systems. It included modern programming constructs like IF-ELSE statements and character string handling.
- **Fortran 90 (1991):** Introduced array programming, module systems, and recursion, aligning Fortran with contemporary programming practices.
- **Fortran 2003 and 2008:** Brought object-oriented programming and parallel computing support, respectively.

## Legacy and Influence

The influence of Fortran extends beyond its direct usage. Many modern languages, including Python, C, and even some features of modern-day Java, have been influenced by concepts introduced by Fortran. Its emphasis on efficient computation and handling complex mathematical operations set a standard that continuously inspires programming language development today.

## Technical Advantages of Fortran

- High performance in numerical computations
- Efficient handling of large datasets
- Optimized for scientific and engineering applications
- Superior array processing capabilities
- Robust support for parallel computing
- Highly mature and stable language with extensive libraries
- Backward compatibility with older Fortran code
- Strong compiler optimizations for speed and efficiency
- Portability across different hardware platforms
- Minimal runtime overhead



# RAZONES PARA ESTUDIAR LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

- 1. Incrementar la capacidad de expresar ideas**
- 2. Mejorar la capacidad de elegir lenguajes apropiados**
- 3. Incrementar la habilidad para aprender nuevos lenguajes**
- 4. Mejorar el entendimiento del significado de la implementación**
- 5. Mejorar el uso de lenguajes que son conocidos**
- 6. Avance general de la computación**

# MEJOR ENTENDIMIENTO DEL SIGNIFICADO DE LA IMPLEMENTACIÓN

- ▶ Los mejores programadores son los que tienen al menos la comprensión de cómo funcionan las cosas por de bajo:  
**Comprender los problemas y detalles de la implementación.**
- ▶ Conocer los detalles de implementación ayuda a usar un lenguaje de manera más inteligente y escribir el código que es más eficiente y permite visualizar cómo una computadora ejecuta construcciones de lenguaje
- ▶ Cierta tipo de problemas pueden ser solucionados sólo cuando se tiene un conocimiento sobre los detalles de la implementación, por ejemplo: causados por una saturación de búfer o *aliasing* y hacer trucos en aquellos casos en que sea necesario.



# MEJOR ENTENDIMIENTO DEL SIGNIFICADO DE LA IMPLEMENTACIÓN

- ▶ Afecta el diseño del lenguaje, es decir, qué construcciones se incluyen y excluyen. Por ejemplo:

FORTRAN fue diseñado para ser rápido; IBM 704 tenía tres registros de direcciones, por lo que las matrices se limitaban a no ser más que tridimensionales.

- ▶ ¿por qué hay tipos separados para enteros y reales?
- ▶ ¿por qué las "matrices asociativas" se han vuelto comunes como construcciones integradas solo en lenguajes introducidos recientemente?
- ▶ ¿Tiene que ver con la complejidad de la implementación?

# RAZONES PARA ESTUDIAR LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

1. **Incrementar la capacidad de expresar ideas**
2. **Mejorar la capacidad de elegir lenguajes apropiados**
3. **Incrementar la habilidad para aprender nuevos lenguajes**
4. **Mejorar el entendimiento del significado de la implementación**
5. **Mejorar el uso de lenguajes que son conocidos**
6. **Avance general de la computación**

# MEJORAR EL USO DE LENGUAJES QUE SON CONOCIDOS

- ▶ Muchos lenguajes son grandes y complejos
  - ▶ Es poco común estar en conocimiento de todas las características
- ▶ Los programadores sólo están familiarizados con las características que estos utilizan.
- ▶ Estudiando los conceptos de lenguajes de programación los programadores pueden aprender más fácilmente sobre partes que anteriormente no conocían

# RAZONES PARA ESTUDIAR LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

- 1. Incrementar la capacidad de expresar ideas**
- 2. Mejorar la capacidad de elegir lenguajes apropiados**
- 3. Incrementar la habilidad para aprender nuevos lenguajes**
- 4. Mejorar el entendimiento del significado de la implementación**
- 5. Mejorar el uso de lenguajes que son conocidos**
- 6. Avance general de la computación**

# AVANCE GENERAL DE LA COMPUTACIÓN

- ▶ Nuevas formas de pensar sobre la informática, nueva tecnología, por lo tanto, necesidad de nuevos conceptos de lenguaje apropiados.
- ▶ La historia: A veces el desconocimiento de ciertas cuestiones sobre los lenguajes de programación influyen en la aceptación de otros nuevos
- ▶ Ejemplos: Algol 60, Smalltalk
- ▶ En general con un mayor conocimiento de los conceptos de LPs, la industria haría un mejor trabajo al adoptar lenguajes basados en sus méritos en lugar de las fuerzas políticas o de otro tipo.
- ▶ Por ejemplo, Algol 60 nunca fue popular a pesar de ser superior a FORTRAN

# RAZONES PARA ESTUDIAR LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

- 1. Incrementar la capacidad de expresar ideas**
- 2. Mejorar la capacidad de elegir lenguajes apropiados**
- 3. Incrementar la habilidad para aprender nuevos lenguajes**
- 4. Mejorar el entendimiento del significado de la implementación**
- 5. Mejorar el uso de lenguajes que son conocidos**
- 6. Avance general de la computación**

# HABILIDADES PARA DESARROLLAR UN LENGUAJE PROPIO

- ▶ Es posible que deba diseñar un lenguaje de propósito especial para ingresar los comandos para un software que desarrolle.
  - Un lenguaje para una interfaz de robótica
- ▶ Estudiar los conceptos de PL le dará la capacidad de diseñar un nuevo lenguaje adecuado y eficiente para sus requisitos de software.

# EL USO EN LOS ÚLTIMOS 25 AÑOS

Programming Language	2024	2019	2014	2009	2004	1999	1994	1989
Python	1	3	8	6	8	26	23	-
C	2	2	1	2	2	1	1	1
C++	3	4	4	3	3	2	2	2
Java	4	1	2	1	1	16	-	-
C#	5	6	5	7	7	21	-	-
JavaScript	6	7	9	9	9	18	-	-
Visual Basic	7	19	236	-	-	-	-	-
SQL	8	9	-	-	92	-	-	-
Go	9	17	36	-	-	-	-	-
PHP	10	8	6	4	6	-	-	-
Objective-C	33	10	3	33	41	-	-	-
Lisp	34	32	14	20	14	14	6	3
(Visual) Basic	-	-	7	5	5	3	3	7



# RELACIÓN CON OTRAS ÁREAS

- ▶ Áreas usuales de estudio: lenguajes de programación, compiladores, arquitectura de computadoras, sistemas operativos, bases de datos, ingeniería de software, computación gráfica
- ▶ Muchas cosas interesantes ocurren en los límites:
  - ▶ Desarrollo de procesadores – arquitectura de comp. y compiladores
  - ▶ Sistemas operativos – kernel y las librerías de tiempo de ejecución
  - ▶ Lenguajes de programación – compilador y librerías de tiempo de ejecución
  - ▶ Sistemas de memoria – sistemas operativos, hardware, compiladores
  - ▶ Diseño de lenguajes – cuestiones de implementación

# IMPORTANCIA DEL ESTUDIO DE LAS ESTRUCTURAS DE LOS LENGUAJES DE PROGRAMACIÓN

Los **lenguajes de programación** tienen una fuerte **influencia** **la manera de formular las soluciones a los problemas**

Los **paradigmas** poseen estilos de programación muy diferentes, e **influyen la manera en la cual los programadores ven los algoritmos**

Conocimiento del cómo funcionan los lenguajes de programación facilita aprender nuevos lenguajes y utilizar adecuadamente lenguajes existentes.

# INFLUENCIAS PRINCIPALES EN EL DISEÑO DE LENGUAJES

- ▶ **Arquitectura de computación:** nosotros utilizamos lenguajes imperativos, al menos en parte debido a que utilizamos máquinas de von Neumann
- ▶ **Metodologías de programación:**
  - ▶ 50s y comienzos de los 60s: aplicaciones simples, preocupación con relación a la eficiencia de máquina;
  - ▶ Fines de los 60s: La eficiencia de las personas se vuelve visiblemente más importante; legibilidad, mejores estructuras de control
  - ▶ Fines de los 70s: Abstracción de datos
  - ▶ Mediados de los 80s: Programación orientada a objetos

# CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN



# ¿PORQUÉ EXISTEN TANTOS LENGUAJES?

- ▶ **Evolución:** búsqueda continua de hacer mejor las cosas
- ▶ **Propósito especial:** lenguajes diseñados con un dominio de problemas específico
- ▶ **Preferencia personal:** la diferencia en las preferencias hace poco probable que exista *un* lenguaje universalmente aceptado



# DOMINIOS DE PROGRAMACIÓN

## **Aplicaciones científicas:**

Gran número de computaciones en punto flotante, uso de arrays - Fortran

## **Aplicaciones de negocio:**

Producir reportes, uso de números decimales y caracteres - COBOL

## **Inteligencia artificial:**

Se manipulan símbolos en vez de números, uso de listas ligadas - LISP

## **Programación de sistemas:**

Se necesita eficiencia debido a su uso continuo - C

## **Software Web:**

Conjunto ecléctico de lenguajes: markup (ej., HTML), scripting (ej., PHP), propósito general (ej., Java)

# ABSTRACCIONES EN LOS LENGUAJES DE PROGRAMACIÓN

- ▶ Abstracción de datos
- ▶ Abstracción de control
  - ▶ básicas, estructuradas, unitarias
- ▶ Clasificación de los lenguajes conforme manipulan estas abstracciones: paradigmas

# CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

- ▶ Los lenguajes de programación pueden clasificarse en familias de acuerdo a su *modelo de computación*:
  - ▶ Declarativos: qué hace la computadora
  - ▶ Imperativos: cómo lo hace
- ▶ Los lenguajes declarativos pueden considerarse de mayor nivel, están más orientados al punto de vista del programador
- ▶ Los lenguajes imperativos predominan por razones de desempeño



# CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

## **Imperativos** (Ada, Pascal, C):

- ▶ Por lejos el más común y familiar
- ▶ El modelo de computación se basa en la modificación de variables
- ▶ Basados en sentencias (principalmente asignación) que influyen los cálculos subsiguientes

## **Funcionales** (Lisp, Haskell)

- ▶ Basados en la definición y aplicación recursiva de funciones
- ▶ Toman su inspiración del cálculo lambda (Church, 1930)
- ▶ Un programa se considera una función de entradas a salidas, definidas en términos de funciones más simples a través de un proceso de refinamiento

# CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

## **Orientados a objetos** (Smalltalk, C++, Java, C#):

- ▶ Raíces en Simula67, están relacionados con el modelo de von Neumann pero con un modelo más estructurado de memoria y computación
- ▶ En vez de ver a la computación como la operación de un procesador monolítico, los OOL la ve como interacciones entre objetos semindependientes cada uno de los cuales tiene su propio estado interno y funciones para manipular el estado

# MÉTODOS DE IMPLEMENTACIÓN

## 1. Compilación

- Traducción de un programa escrito en lenguaje de alto nivel a código máquina
- Traducción lenta
- Ejecución rápida

## 2. Interpretación pura

- Sin traducción
- Ejecución lenta
- Cada vez más rara

## 3. Sistemas de implementación híbrida

- Bajo costo de traducción
- Velocidad de ejecución media

# BIBLIOGRAFIA

Concepts of Programming Languages, 11/E. Robert W. Sebesta, ISBN: 978-0-13-394302-3. 2016. Pearson.

