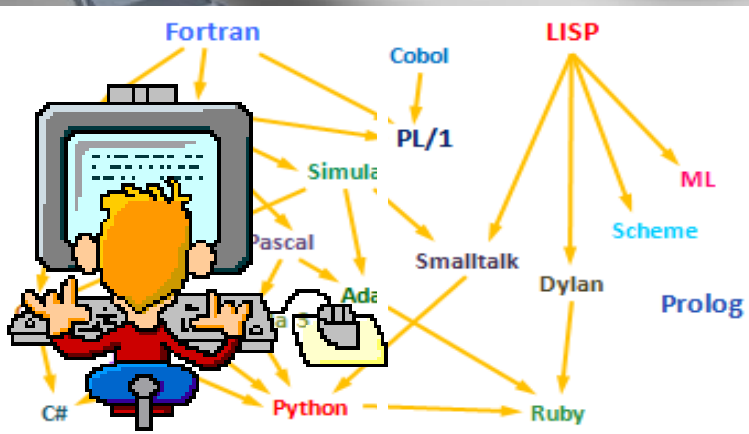




Programación I

Ing. Carlos R. Rodríguez

***Tecnicatura
Universitaria en
Programación***



UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Mendoza



Programación I

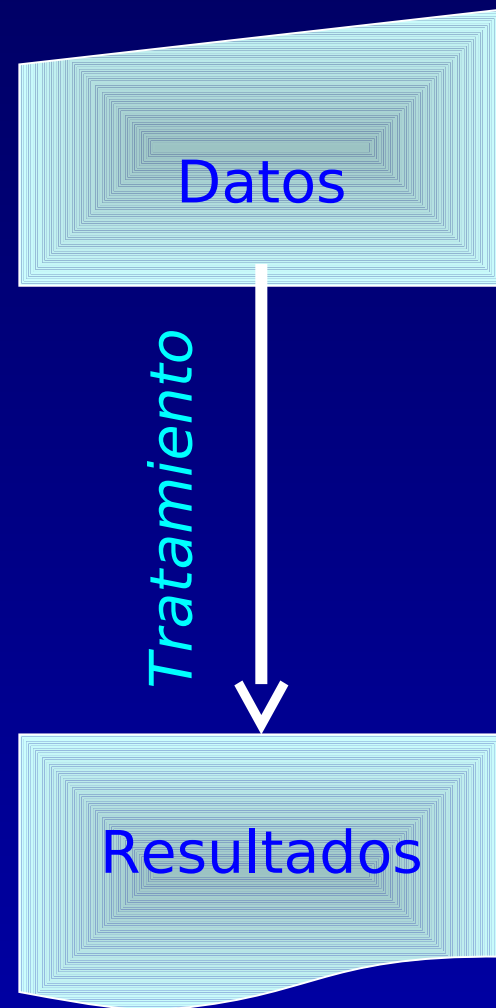
Con pseudocódigo





Conclusiones de la *clase anterior*:

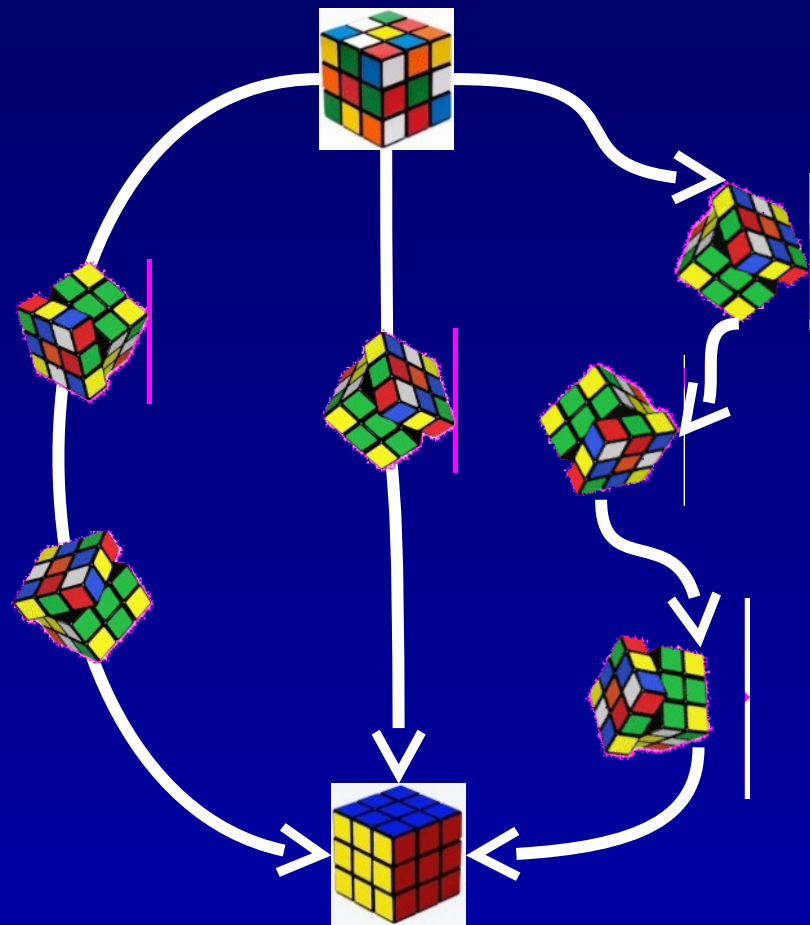
- Un algoritmo es un camino *seguro* entre los **datos** del **problema** y su **solución**, mediante *cierto tratamiento* de los primeros.
- Siempre tiene una **condición de finalización** que debe cumplirse luego de un número **finito** de **pasos**.
- A partir de cero o más **datos** produce **uno o más resultados**.





Conclusiones de la *clase anterior*:

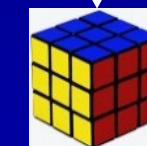
- El **camino** *no es necesariamente único* y *pueden existir múltiples soluciones*.
- Dos o más soluciones que partiendo de un mismo punto o estado inicial producen un mismo resultado se denominan **equivalentes**.





Conclusiones de la *clase anterior*:

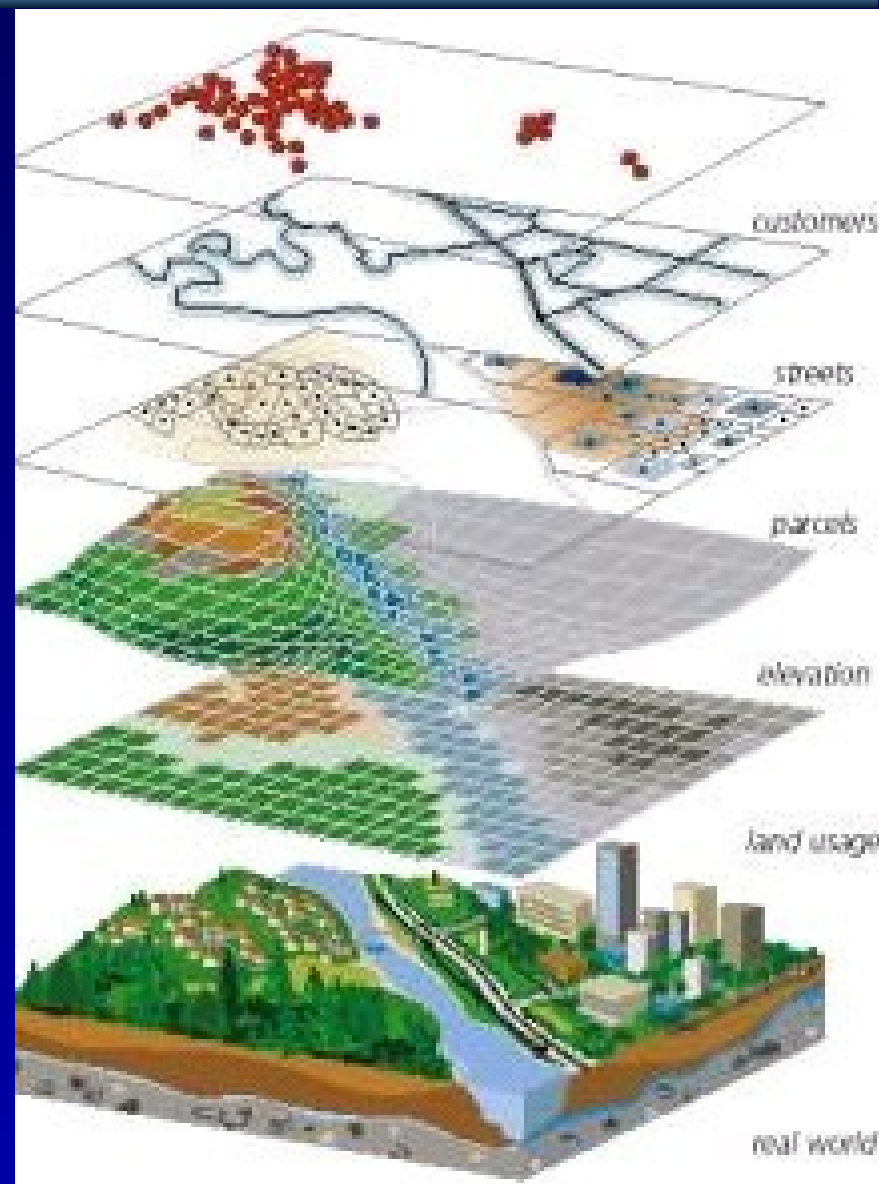
- Cuando hay más de una **solución posible**:
 - Hay soluciones mejores y peores.
 - ▮ La **mejor solución** se conoce como **óptima** y consume la **menor cantidad** de recursos:
 - **Tiempo** de *ejecución*.
 - ▮ **Memoria** *necesaria*.
 - ▮ Suele ser la más *simple*.





La abstracción es una operación mental de *ignorancia selectiva voluntaria* por la cual desechamos lo **accesorio** – *dependiente del problema* – y nos quedamos con lo **esencial**.

Todo lenguaje de programación es una abstracción del *ambiente de ejecución* del programa.

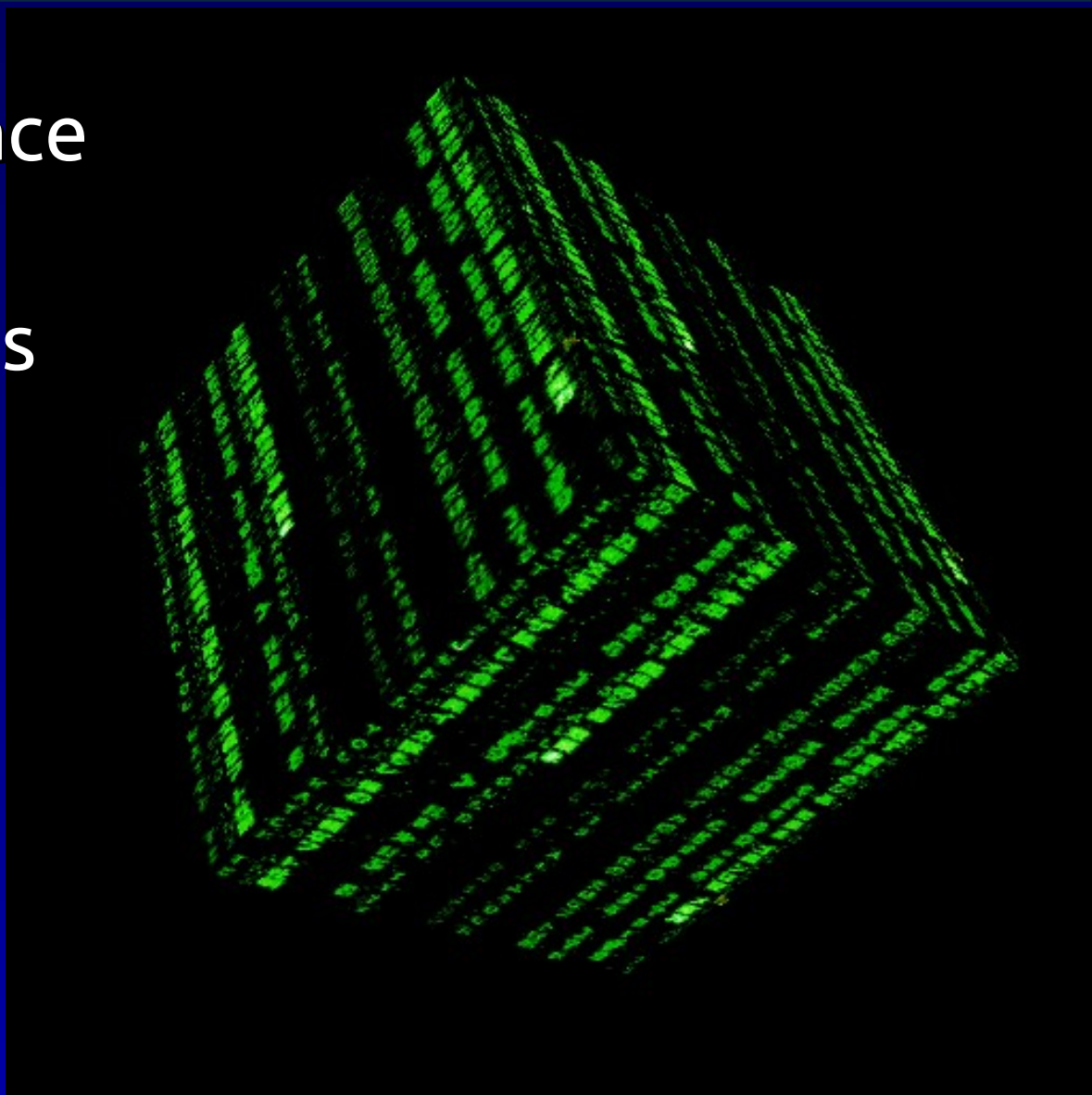


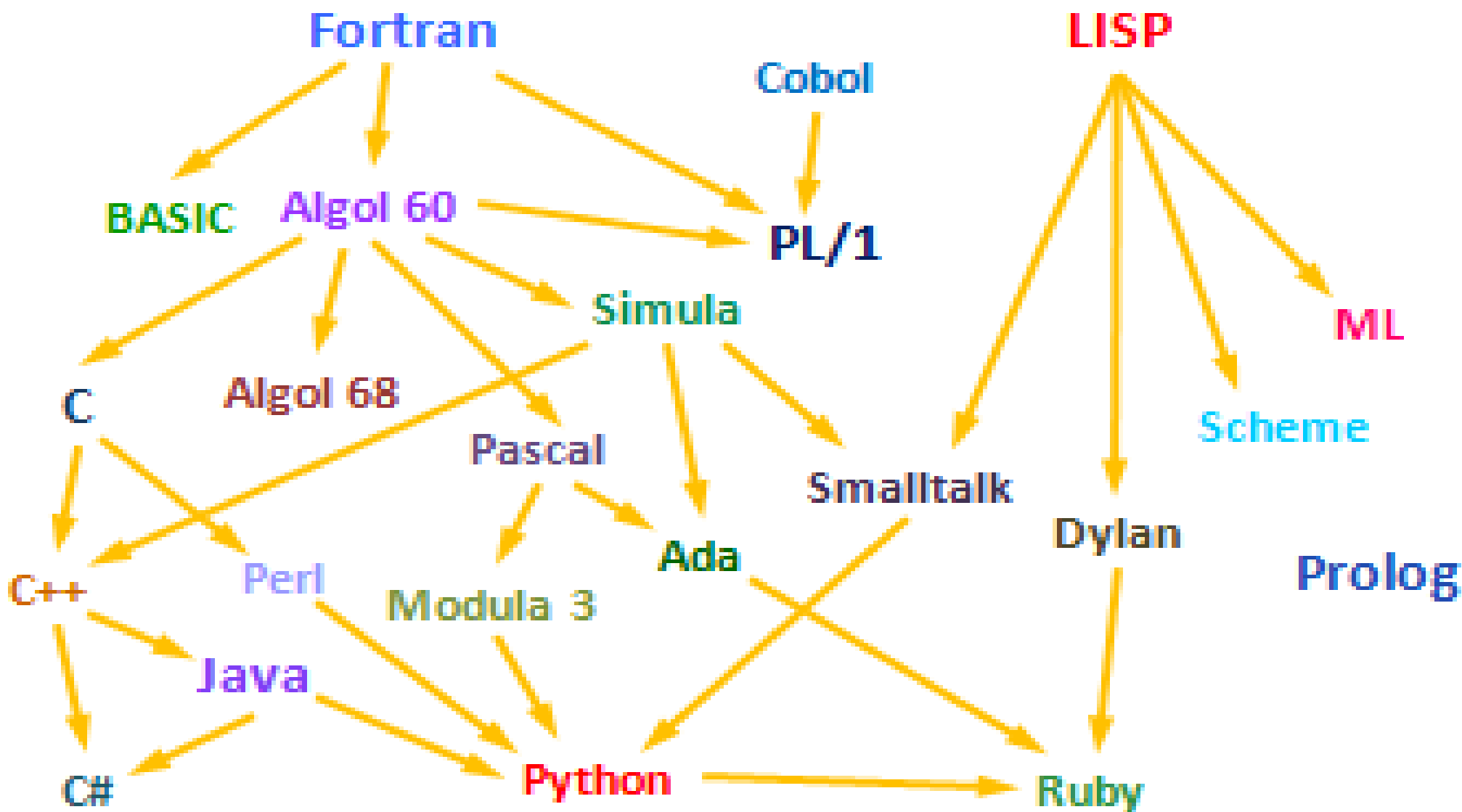


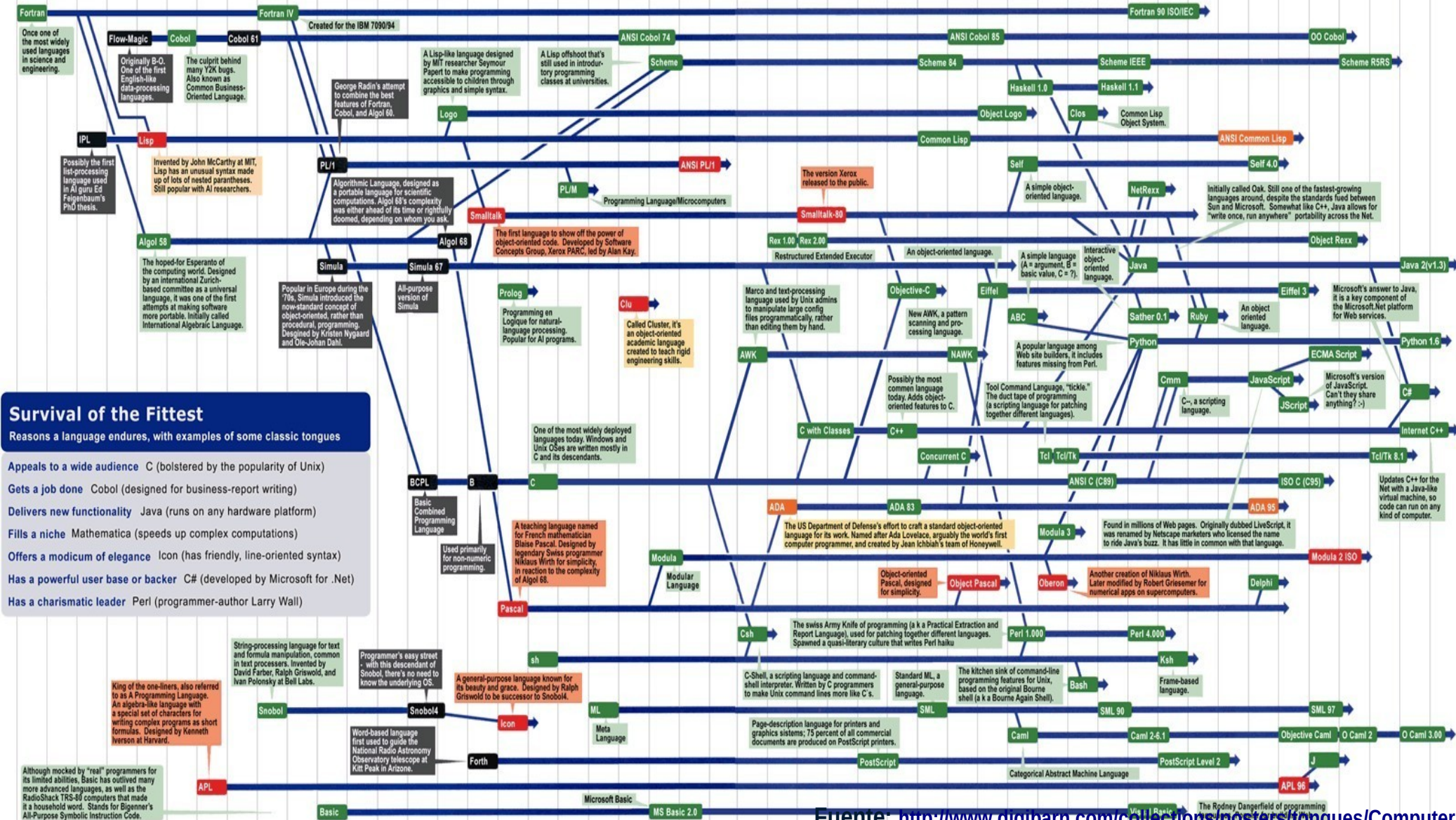
Existen lenguajes de programación desde hace más de medio siglo.

En la actualidad hay más de 2500 lenguajes de programación *documentados*.

La gran mayoría descende *directa* o *indirectamente* de los 3 lenguajes originales.


















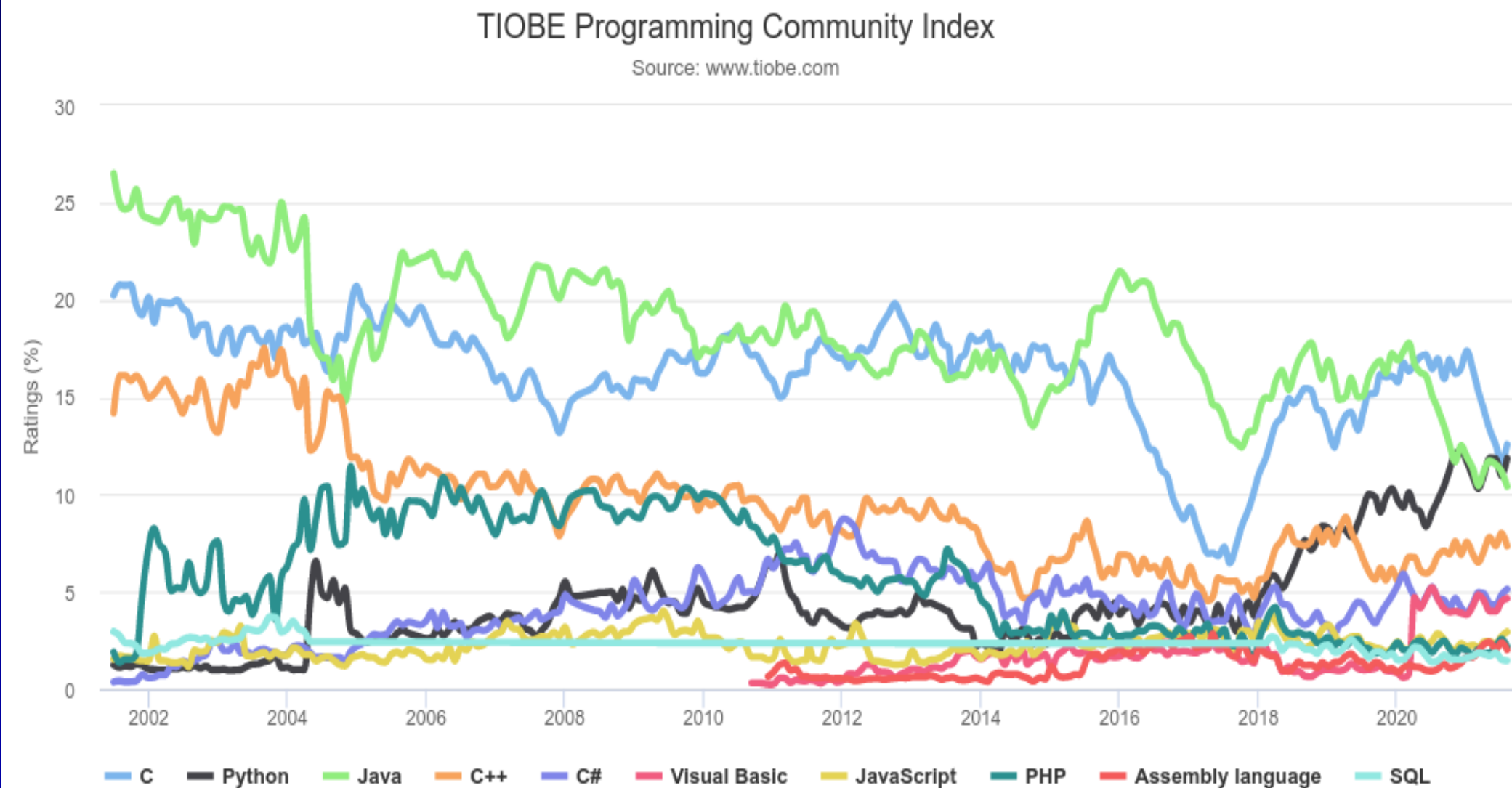
Respecto al uso de cada uno, la situación hoy es:

Aug 2021	Aug 2020	Change	Programming Language		Ratings	Change
1	1			C	12.57%	-4.41%
2	3	^		Python	11.86%	+2.17%
3	2	v		Java	10.43%	-4.00%
4	4			C++	7.36%	+0.52%
5	5			C#	5.14%	+0.46%
6	6			Visual Basic	4.67%	+0.01%
7	7			JavaScript	2.95%	+0.07%
8	9	^		PHP	2.19%	-0.05%
9	14	^^		Assembly language	2.03%	+0.99%
10	10			SQL	1.47%	+0.02%

Fuente: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>



Y su evolución histórica ha sido:





Y su evolución histórica de largo plazo ha sido:

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Java	2	1	1	1	3	18	-	-
Python	3	5	6	8	26	24	-	-
C++	4	3	3	3	2	2	2	6
C#	5	4	5	7	13	-	-	-
Visual Basic	6	13	-	-	-	-	-	-
JavaScript	7	7	10	9	9	22	-	-
PHP	8	6	4	4	10	-	-	-
SQL	9	-	-	-	37	-	-	-
Assembly language	10	11	-	-	-	-	-	-
Ada	31	27	17	17	18	8	5	2
Lisp	34	28	13	13	16	7	8	3
(Visual) Basic	-	-	7	5	4	3	3	5

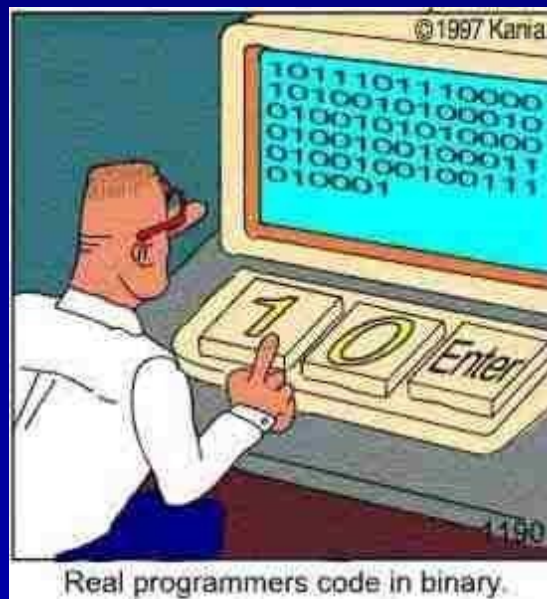


Las computadoras no son sólo *máquinas* sino *herramientas* que *amplifican nuestro pensamiento*.

Como resultado, estas herramientas *empiezan a parecerse* menos a las máquinas y *más a partes de nuestras mentes*.

“La génesis de la revolución de las computadoras fue dentro de una máquina. Así, la génesis de nuestros lenguajes de programación tendieron a parecerse a aquella máquina”.

Bruce Eckel





Un paradigma de programación representa un **enfoque** particular o **filosofía** para la construcción del software.

Algunos ejemplos son:

- El paradigma **imperativo** **procedural** es muy común aún y está representado por **C**, **Pascal** y **BASIC**, entre otros.

```
#include <stdio.h>

int main (void)
{
    int n,x=0,y=1,i,z;
    print("\tSerie de Fibonacci");
    print("\nEscriba la cantidad de numeros que quiere de la serie:");
    scanf("%d", &n);
    print("%d,%d,", x,y);

    for (i=3;i<=n;i++)
    {
        z=x+y;
        x=y;
        y=z;
        print("%d,", z);
    }
    return 0;
}
```




- El paradigma **funcional** está representado por la familia de lenguajes **LISP** (en particular **Scheme**), **ML** o **Haskell**.

fib :: Integer -> Integer
fib 0 = 0
fib 1 = 1
fib n = **fib (n-1)** + **fib (n-2)**



- El paradigma **lógico**, un ejemplo es **PROLOG**.

```
my_fibonacci(_, _, 0).
```

```
my_fibonacci(A, B, N):-
```

```
    write(A),
```

```
    write(' '),
```

```
    N1 is N-1,
```

```
    C is A+B,
```

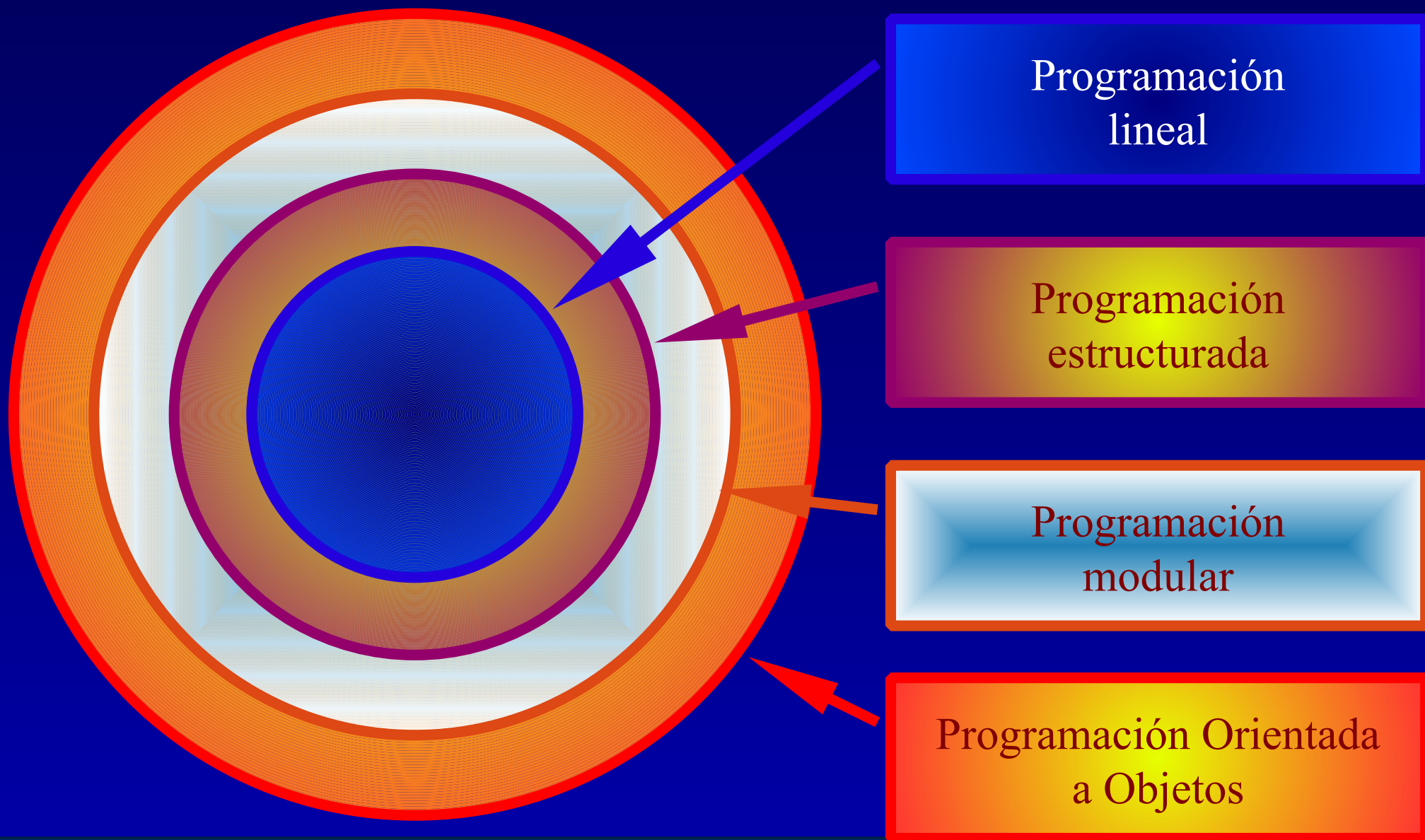
```
    my_fibonacci(B, C, N1).
```



- El paradigma de Programación Orientada a Objetos. Un lenguaje completamente orientado a objetos es Smalltalk.

Smalltalk

```
^self <= 2
  ifTrue: [1]
  ifFalse: [(self - 1) fibonacci + (self - 2) fibonacci]
```





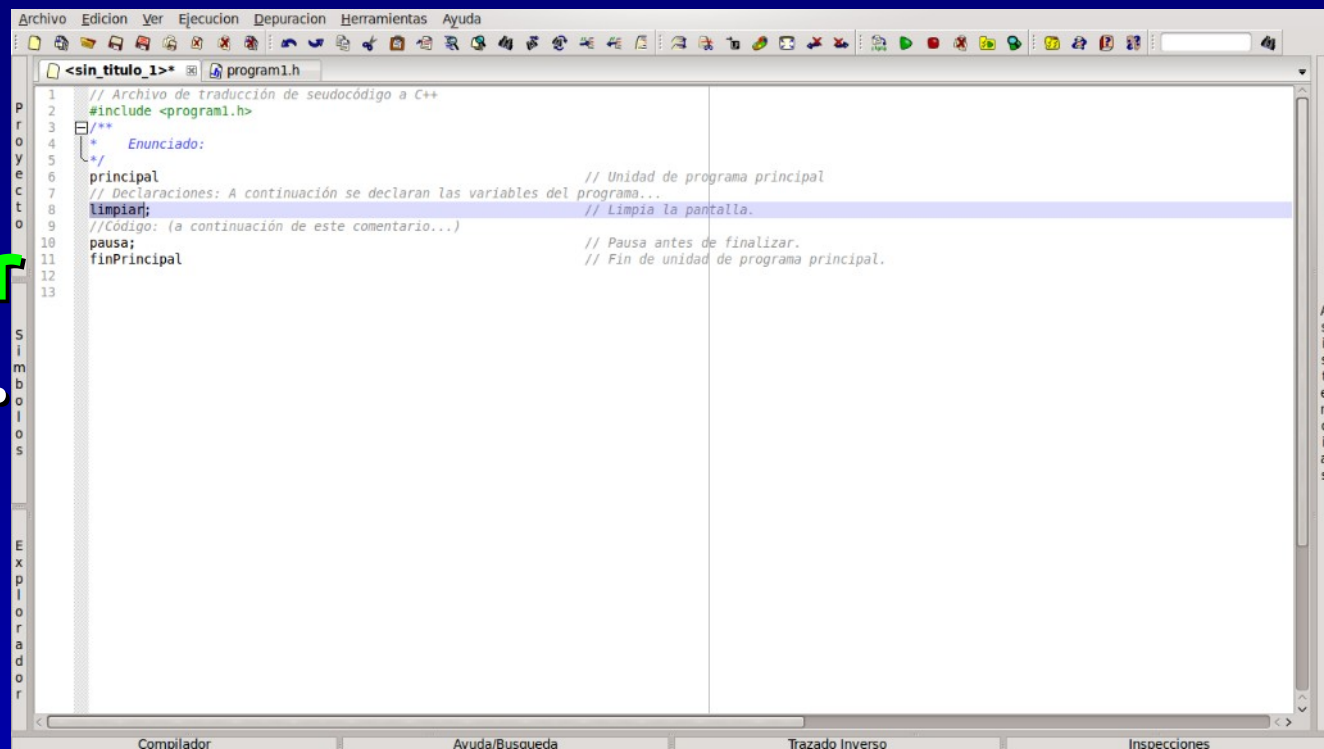
Cualquiera sea el lenguaje y la plataforma sobre la que se ejecute, existe un **ambiente** que provee:

- Un **editor** de *código fuente*.

- Un **traductor** (*compilador o intérprete*).

- Un **constructor** de *aplicaciones*.

- Un **depurador** de *errores*.



```
1 // Archivo de traducción de pseudocódigo a C++
2 #include <program1.h>
3
4 /**
5  * Enunciado:
6  */
7 principal
8 // Declaraciones: A continuación se declaran las variables del programa...
9 limpiar; // Limpia la pantalla.
10 //Código: (a continuación de este comentario...)
11 pausa; // Pausa antes de finalizar.
12 finPrincipal; // Fin de unidad de programa principal.
13
```