



**UNIVERSIDAD NACIONAL DE UCAYALI**  
**FACULTAD DE INGENIERIA DE SISTEMAS E INGENIERIA CIVIL**  
**ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS**



---

## **PROGRAMACIÓN**

---

**CURSO** : PRACTICAS PRE-PROFESIONALES 1

**CICLO** : VII

**DOCENTE** : ING. DIAZ ESTRADA, DIANA MARGARITA

**ALUMNOS:**

- LÓPEZ MONTALVÁN, HÉCTOR ANDREY.
- MENDOZA FLORES, JAMT AMERICO.
- ONUKI NAVAS, AURELIO YOSHIO.
- SUAREZ MACIEL, SUSANA ISABEL.

**PERÚ**

**2022**

## Contenido

Introducción .....	4
Historia .....	6
INTERPRETES Y COMPILADORES .....	7
FORTRAN.....	8
COBOL .....	8
BASIC .....	9
LOGO.....	10
C .....	11
PASCAL.....	12
JAVA.....	13
¿Qué es la programación?.....	14
Tipos de programación .....	14
1. Programación estructurada (PE) .....	14
2. Programación modular .....	15
3. Programación orientada a objetos (POO).....	15
4. Programación concurrente .....	16
5. Programación funcional.....	16
6. Programación lógica.....	16
Habilidades que buscan las empresas en un desarrollador .....	17
Habilidades de un desarrollador Web .....	18
Áreas de la programación.....	18
1. Desarrollo Web .....	18

2. Desarrollo móvil .....	19
3. Videojuegos .....	20
4. Realidad virtual y aumentada .....	20
5. Desarrollo de aplicaciones de escritorio .....	21
6. Sistemas Operativos / Embebidos.....	21
7. Seguridad informática .....	21
8. Machine Learning.....	21
9. Cloud Computing .....	22
Conclusión.....	24

## Introducción

Las personas para comunicarse entre sí utilizan un lenguaje que puede ser oral o escrito. En general, para comunicar algo siempre se usa un lenguaje.

La informática no queda excluida del uso de lenguajes, ya que estos son la manera de especificar las acciones que se desea sean realizadas en la computadora.

En otras palabras, son las interfaces entre el programador y la computadora. A través de ellos podemos desarrollar programas o aplicaciones, que se componen por un conjunto de instrucciones que luego se ejecutarán en la computadora haciendo uso de sus recursos (CPU, memoria, disco, etc.).

Los lenguajes de programación están destinados a distintos ámbitos, dependiendo de sus características que simplifican algunas tareas y complejizan otras.

Pueden estar destinados a aplicaciones científicas, aplicaciones de negocios, inteligencia artificial, programación de sistemas, scripting, y también disponemos de lenguajes de propósitos especiales.

Los lenguajes de programación tienen una estructura compleja que se compone de varias partes: sintaxis, semántica, elementos del lenguaje, nivel de abstracción, paradigma, estructuras de control para ordenar la ejecución de los programas, tipos de datos (números, letras, etc.), y funciones o procedimientos (unidades) que contienen un conjunto de instrucciones, entre otras.

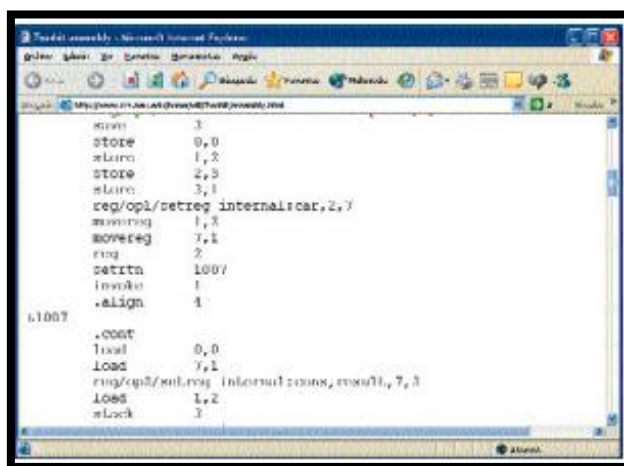
No hay un único tipo de lenguajes, sino que se clasifican según las características que posean y según el paradigma y conceptos que soporten.

Lenguajes de programación hay en gran cantidad, algunos han evolucionado a lo largo del tiempo y siguen vigentes en el transcurso de muchos años, mientras que otros han sido operativos durante un periodo más o menos largo y actualmente no se usan. Dada esta gran variedad de lenguajes, no se pretende dar una visión de todos, sino una clasificación en diversos tipos y concretar algunos de ellos. En general un lenguaje es

un método conveniente y sencillo de describir las estructuras de información y las secuencias de acciones ejecutar una tarea concreta.

## Historia

En los primeros tiempos de la informática, la programación se efectuaba en el único lenguaje que entiende el microprocesador: su propio código binario, también denominado lenguaje máquina o código máquina. Pero la programación en lenguaje máquina resulta muy lenta y tediosa, pues los datos e instrucciones se deben introducir en sistema binario y, además, obliga a conocer las posiciones de memoria donde se almacenan los datos. Como puede imaginar, este tipo de programación conlleva gran número de errores y la tarea de depuración exige bastante tiempo y dedicación. Por este motivo, a principios de los 50 se creó una notación simbólica, denominada código de ensamblaje (ASSEMBLY), que utiliza una serie de abreviaturas mnemotécnicas para representar las operaciones (Ilustración 1): ADD (sumar), STORE (copiar), etc. Al principio, la traducción del código de ensamblaje al código máquina se realizaba manualmente, pero enseguida se vio que el ordenador también podía encargarse de esa traducción; se desarrolló así un programa traductor, llamado ensamblador (ASSEMBLER).



*Ilustración 1: Ejemplo de programación en código de ensamblaje*

Conforme los ordenadores fueron introduciéndose en el mundo empresarial y académico, aquellos primitivos lenguajes fueron sustituidos por otros más sencillos de aprender y más cómodos de emplear. Estos lenguajes, llamados de alto nivel, tienen una estructura que se adapta más al pensamiento humano que a la forma de trabajar

del ordenador. Por ejemplo, seguro que le suenan lenguajes como BASIC, PASCAL, C, etc.

## **INTERPRETES Y COMPILADORES**

Imagine que no sabe nada de inglés y necesita conversar con alguien que solo conoce ese idioma. La forma más sencilla de establecer comunicación es conseguir una persona que ejerza de intérprete. Cuando diga una frase en castellano, su intérprete la traducir al inglés y, de esta forma, podrá entenderla aquella persona con la que esté conversando; análogo proceso se seguirá para traducir del inglés al castellano. En resumen, mientras esté presente su intérprete, la conversación es posible. El intérprete informático realiza, más o menos, el mismo papel. Traduce instrucción a instrucción y, de esta forma, favorece la interactividad, la depuración y puesta a punto del programa, la ejecución inmediata de una orden, etc. Por ejemplo, entre los lenguajes que suelen ser interpretados, se encuentran BASIC, LOGO, etc.

El equivalente informático de esta modalidad de traductor se denomina compilador. Observe que, en contraste con el intérprete, que traduce las instrucciones una a una, el compilador traduce todo el programa de golpe, dejándolo listo para ser ejecutado. De esta forma, se logra mayor rapidez en la ejecución y, además, se liberan recursos de la memoria, pues el programa, una vez compilado, no exige que el traductor esté residente en memoria, como sucede con los intérpretes. Por ejemplo, entre los lenguajes que siempre son compilados se pueden destacar PASCAL, FORTRAN, COBOL, etc.

Sin embargo, no todo son ventajas en los lenguajes compilados. Así, la depuración del programa resulta más cómoda con un intérprete, ya que el compilador no informa de los posibles errores hasta el momento de la compilación. Por otra parte, cada vez que se modifica algo en el programa es preciso volver a compilarlo de nuevo.

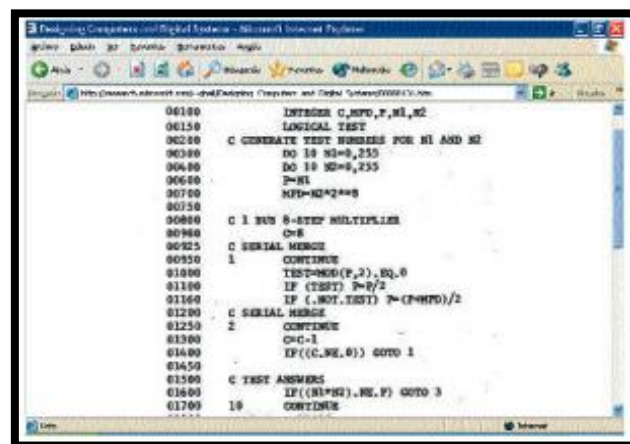
Una vez aclarada la diferencia entre intérprete y compilador, vayamos ya con el primer lenguaje de alto nivel de amplia difusión.

## FORTRAN

Al comienzo de la década de los 50, John Backus estaba trabajando con SSEC (Selective Sequence Electronic Calculator), uno de los primeros ordenadores de IBM, y desarrolló el programa SPEEDCODING para Él. Tomando Éste como base, se emprendió, en otoño de 1954, la creación de un lenguaje para añadirle más prestaciones al modelo IBM 704, que iba a salir pronto al mercado.

En 1956 se terminó el compilador FORTRAN (FORMula TRANslator) y se incluyó en el IBM 704, junto con un manual de 51 páginas

Como su nombre indica, FORTRAN estaba (y está-) destinado a la resolución de problemas científico-técnicos, resultando relativamente sencillo de aprender si se domina la notación matemática.



*Ilustración 2: Programa en FORTRAN*

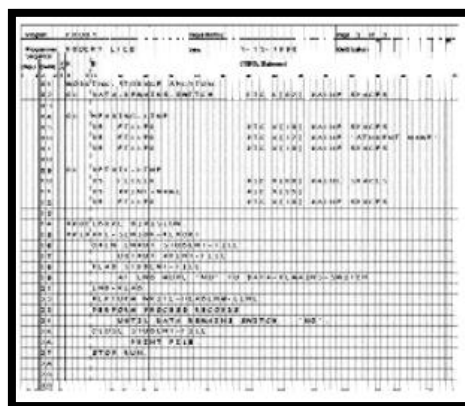
## COBOL

A finales de los 50, el Departamento de Defensa USA estaba bastante preocupado con los lenguajes de programación existentes, especialmente por dos razones: los programas no podían llevarse de un ordenador a otro y resultaban bastante difíciles de leer y modificar. Para solventar estos inconvenientes, patrocinó una conferencia sobre lenguajes (CODASYL, CONference on DATA SYstems Languages), que tuvo lugar en



1959 y en la que participaron las grandes empresas del sector (IBM, Sperry Rand, Honey Well, etc.). Como era previsible, formaba parte del comité la gran dama de la informática, Grace Hopper.

Fruto de aquella conferencia fueron las especificaciones para desarrollar COBOL (COmmon Business Oriented Language), un lenguaje orientado hacia funciones administrativas, de gran portabilidad y legibilidad. Su primera versión apareció al año siguiente y, con el paso de los años, surgieron nuevas actualizaciones: COBOL 74, COBOL 85, etc. Ya que se buscaba su facilidad de lectura, COBOL tiene una sintaxis muy similar al inglés común (figura 3), cuya terminología aparece continuamente: verbos, párrafos, frases, etc. Así, los programas se estructuran en cuatro divisiones (Identification, Environment, Data, Procedure), que se subdividen en secciones y estas, a su vez, en párrafos, que constan de frases e instrucciones.



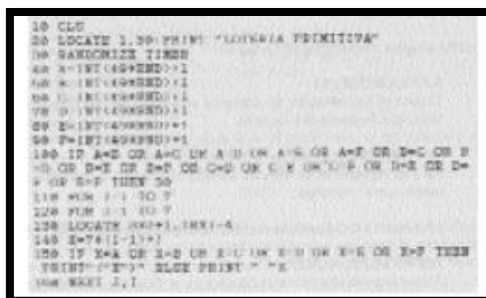
*Ilustración 3: Antigua hoja de codificación en COBOL*

En la actualidad, COBOL se utiliza casi exclusivamente en algunos grandes sistemas informáticos (entidades bancarias, sobre todo), si bien más para mantener el código existente que para desarrollar nuevas aplicaciones.

## **BASIC**

John G. Kemeny y Thomas E. Kurtz eran profesores del Dartmouth College (New Hampshire) y, en 1964, diseñaron un nuevo lenguaje que permitiera introducirse a sus estudiantes en los sistemas de tiempo compartido. Ese lenguaje, al que llamaron BASIC

por su sencillez, es, sin duda, el más difundido, aplicándose tanto en tareas de gestión como en aplicaciones científicas (figura 4).



```
10 CLS
20 LOCATE 1,30:PRINT "JUEGO PRIMITIVA"
30 RANDOMIZE TIMER
40 A=INT(60/255)*255
50 B=INT(60/255)*255
60 C=INT(60/255)*255
70 D=INT(60/255)*255
80 E=INT(60/255)*255
90 F=INT(60/255)*255
100 IF A=0 OR A=1 OR A=2 OR A=3 OR A=4 OR A=5 OR A=6 OR A=7 OR A=8 OR A=9 THEN 30
110 IF B=0 OR B=1 OR B=2 OR B=3 OR B=4 OR B=5 OR B=6 OR B=7 OR B=8 OR B=9 THEN 30
120 IF C=0 OR C=1 OR C=2 OR C=3 OR C=4 OR C=5 OR C=6 OR C=7 OR C=8 OR C=9 THEN 30
130 IF D=0 OR D=1 OR D=2 OR D=3 OR D=4 OR D=5 OR D=6 OR D=7 OR D=8 OR D=9 THEN 30
140 IF E=0 OR E=1 OR E=2 OR E=3 OR E=4 OR E=5 OR E=6 OR E=7 OR E=8 OR E=9 THEN 30
150 IF F=0 OR F=1 OR F=2 OR F=3 OR F=4 OR F=5 OR F=6 OR F=7 OR F=8 OR F=9 THEN 30
160 PRINT "FIN" ELSE PRINT " " GOTO 10
170 WAIT 1,1
```

*Ilustración 4: Programa en BASIC que simula un sorteo*

Microsoft adaptó su BASIC a los productos de Apple, a los microordenadores y, lo más importante, al PC de IBM; de hecho, el sistema operativo MSDOS incluía la versión GW-BASIC. En resumen, mucha gente aprendió a programar en BASIC con su ZX-Spectrum o su primer PC y, una vez dominado un lenguaje, es comprensible una cierta reticencia al cambio.

BASIC ha sabido adaptarse a las necesidades del mercado en el transcurso de los años. Así, las primeras versiones eran interpretadas y sus programas resultaban un tanto ilegibles; en cambio, las actuales incorporan bastante estructuración y son compiladas. El exponente máximo de los modernos BASIC es Visual BASIC.

## LOGO

En 1964, Seymour Papert se incorporó al MIT, tras haber permanecido cinco años en Suiza, colaborando con el pedagogo Jean Piaget (1896-1980). Tres años después, Papert comenzó a diseñar un lenguaje que sirviera para introducir en el mundo de la programación al alumnado de menor edad ¡Que los niños programen a los ordenadores y no los ordenadores a los niños!

Poco a poco, LOGO fue poniéndose a punto y cuando, en 1980, Papert lo divulgó en todo el mundo con su libro *Mindstorms: Children Computers and Powerful Ideas*, fue

muy bien acogido en los ámbitos educativos, especialmente en enseñanza primaria y secundaria.

Tras un impulso inicial muy ilusionante, LOGO ha ido desapareciendo de los centros de enseñanza españoles. Por un lado, resulta que el lenguaje LOGO no es nada sencillo cuando se quiere ir más allá de la tortuga gráfica, ya que se basa en la utilización continua de listas y procedimientos recursivos, que no son fáciles de manejar. Por otra parte, la informática educativa ha ido perdiendo su componente formativa y creativa (la programación) y ha sido sustituida por una informática de usuario (manejo de aplicaciones ofimáticas).

## C

En los Laboratorios Bell (New Jersey) trabajaron dos de los investigadores más conocidos de la moderna informática, Kenneth Thompson y Dennis Ritchie , creadores del sistema operativo UNIX, en 1969.

En 1970, Thompson desarrolló un lenguaje experimental, al que llamó B. Dos años después, Ritchie se basó en B para crear un nuevo lenguaje de propósito general, que denominó C. Como no depende de la arquitectura del hardware, C es uno de los lenguajes más portables del mercado y, como además ofrece amplias prestaciones, su difusión es amplísima.

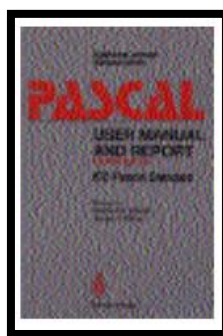
A principios de los 80, Bjarne Stroustrup (figura 12) diseñó una ampliación de C y, en 1984, la convirtió en un compilador que llamó C++, especialmente enfocado a la programación orientada a objetos.



## **PASCAL**

A principios de los 70, el profesor suizo Niklaus Wirth, del Instituto Politécnico Federal de Zurich, emprendió la creación de un nuevo lenguaje (PASCAL) que permitiera introducirse en la programación de una forma fácil, pero a la vez potente y, sobre todo, siguiendo unas pautas estructuradas. De hecho, PASCAL es el lenguaje más sencillo que posibilita el acceso a la informática teórica: descomposición modular, recursividad, punteros, etc.

PASCAL, que surgió como una derivación de ALGOL (ver 'último apartado), fue definido en el libro "PASCAL - User Manual and Report" (1974), escrito por Kathleen Jensen y Niklaus Wirth . En 1980 sufrió la primera formalización y se estandarizó en 1983. Al poco tiempo, Borland lanzó al mercado su compilador PASCAL, cuyo nombre se precedía con la palabra Turbo, para recalcar su rapidez. Su éxito fue tan grande que vendió casi medio millón de copias de su compilador sólo en 1985.



*Ilustración 6: Portada del libro de Pascal*

Durante más de una década, Turbo PASCAL ha sido sinónimo de PASCAL, pero, por desgracia, en el año 2000, Borland dejó de darle soporte técnico y su presencia es cada día menor en el ámbito de la programación, sobreviviendo a duras penas en el mundo universitario. Sin embargo, en 1995 surgió una nueva versión, DELPHI, que amplía PASCAL a la programación visual e intenta hacerle la competencia a Visual BASIC.

## JAVA

Este lenguaje, hoy en día ampliamente utilizado en Internet, fue desarrollado en 1990 por James Gosling (figura 18), de Sun Microsystems, basándose en C y C++. un lenguaje para Internet cuando, en aquella época, la Red estaba casi circunscrita al ámbito universitario? En realidad, el objetivo de Sun no tenía nada que ver con Internet; era crear un interfaz atractivo e intuitivo para electrónica de consumo (calculadoras, televisión interactiva, etc.).



*Ilustración 7: James Gosling*

Sin embargo, la electrónica de consumo no evolucionó como se esperaba y, durante unos años, el lenguaje de Gosling permaneció aparcado, hasta que Bill Joy (cofundador de Sun) consideró que podía ser interesante para Internet y propuso modificarlo para el nuevo medio. En agosto de 1995, ya con el nombre de JAVA, se presentó en sociedad. A pesar de que JAVA resulta un tanto lento en su ejecución, cada día es más popular. Por un lado, es relativamente sencillo y bastante potente; además, es válido para cualquier plataforma y, sobre todo, muy fiable y seguro, manteniendo alejado a los virus.

## **¿Qué es la programación?**

La programación es el proceso de creación de programas informáticos. Esta definición se puede interpretar de la siguiente manera. La programación no es más que una explicación a la computadora de qué, en qué forma y cómo llegar al usuario. En otras palabras, es una especie de arte de traducir los deseos de una persona al lenguaje de la máquina.

## **Tipos de programación**

Los tipos o técnicas de programación son bastante variados, aunque puede que muchos de los lectores sólo conozcan una metodología para realizar programas. En la mayoría de los casos, las técnicas se centran en programación modular y programación estructurada, pero existen otros tipos de programación.

### **1. Programación estructurada (PE)**

La programación estructurada está compuesta por un conjunto de técnicas que han ido evolucionando, aumentando considerablemente la productividad del programa reduciendo el tiempo de depuración y mantenimiento de este.

Esta programación estructurada utiliza un número limitado de estructuras de control, reduciendo así considerablemente los errores.

Esta técnica incorpora:

- Diseño descendente (top-down): el problema se descompone en etapas o estructuras jerárquicas.
- Recursos abstractos (simplicidad): consiste en descomponer las acciones complejas en otras más simples capaces de ser resueltas con mayor facilidad.
- Estructuras básicas: existen tres tipos de estructuras básicas:
- Estructuras secuenciales: cada acción sigue a otra acción secuencialmente. La salida de una acción es la entrada de otra.

- Estructuras selectivas: en estas estructuras se evalúan las condiciones y en función del resultado de estas se realizan unas acciones u otras. Se utilizan expresiones lógicas.
- Estructuras repetitivas: son secuencias de instrucciones que se repiten un número determinado de veces.

Las principales ventajas de la programación estructurada son:

- Los programas son más fáciles de entender
- Se reduce la complejidad de las pruebas
- Aumenta la productividad del programador
- Los programas queden mejor documentados internamente.

Un programa está estructurado si posee un único punto de entrada y sólo uno de salida, existen de "1 a n" caminos desde el principio hasta el fin del programa y, por último, que todas las instrucciones son ejecutables sin que aparezcan bucles infinitos.

## **2. Programación modular**

En la programación modular consta de varias secciones divididas de forma que interactúan a través de llamadas a procedimientos, que integran el programa en su totalidad.

En la programación modular, el programa principal coordina las llamadas a los módulos secundarios y pasa los datos necesarios en forma de parámetros.

A su vez cada módulo puede contener sus propios datos y llamar a otros módulos o funciones.

## **3. Programación orientada a objetos (POO)**

Se trata de una técnica que aumenta considerablemente la velocidad de desarrollo de los programas gracias a la reutilización de los objetos.

El elemento principal de la programación orientada a objetos es el objeto.

El objeto es un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización.

Un objeto contiene varios datos bien estructurados y pueden ser visibles o no dependiendo del programador y las acciones del programa en ese momento.

El polimorfismo y la herencia son unas de sus principales características y por ello dedicaremos más adelante un artículo exclusivamente a tratar estos dos términos.

En DesarrolloWeb.com hemos publicado anteriormente una explicación de lo que es la programación orientada a objetos.

#### **4. Programación concurrente**

Este tipo de programación se utiliza cuando tenemos que realizar varias acciones a la vez.

Se suele utilizar para controlar los accesos de usuarios y programas a un recurso de forma simultánea.

Se trata de una programación más lenta y laboriosa, obteniendo unos resultados lentos en las acciones.

#### **5. Programación funcional**

Se caracteriza principalmente por permitir declarar y llamar a funciones dentro de otras funciones.

#### **6. Programación lógica**

Se suele utilizar en la inteligencia artificial y pequeños programas infantiles. Se trata de una programación basada en el cálculo de predicados (una teoría matemática que permite lograr que un ordenador basándose en hecho y reglas lógicas, pueda dar soluciones inteligentes).



## Habilidades que buscan las empresas en un desarrollador

- Ser capaz de encontrar el punto intermedio entre el pragmatismo y el perfeccionismo. Habrá momentos en los que será necesario un desarrollo perfecto, otro en el que habrá que ponerse el “mono de trabajo” y sacar adelante un parche rápido.
- Ser capaz de diferenciar entre un pequeño éxito y un gran triunfo. No debes dar por finalizado un proyecto hasta que no lo hayas testeado una y otra vez, por más que parezca funcionar.
- Saber aceptar soluciones alternativas a las que venías trabajando.
- Tener una visión de negocio estratégica. Debes ser capaz de entender lo que la compañía trata de lograr con cada proyecto.
- Saber transmitir ideas técnicas a personas sin estos conocimientos y si, además, después puedes enseñar a otros desarrolladores mejor que mejor.
- Simplificar los problemas en lugar de complicarlos.
- Tener compromiso e implicación. Debes saber planificarte para cumplir los tiempos de entrega.
- Ser proactivo, tener iniciativa y ganas de aprender para ir mejorando día a día. Una ambición sana será tu aliada.
- Estar motivado, ser eficiente y creativo.
- Ser perseverante, tolerar la frustración.
- Que te apasione el código más allá de tu trabajo.
- Tener capacidad de adaptación, lo que hoy sirve, mañana puede quedarse obsoleto.
- Ser un programador Full Stack.
- Tener una base sólida en algún lenguaje orientado a objetos.
- Formación complementaria (por ejemplo, un Bootcamp te serviría de mucho)
- Ser capaz de aprender nuevas tecnologías rápidamente.

- Experiencia en el extranjero y movilidad geográfica.
- Dominar un segundo idioma, mínimo saber defenderse en inglés y poder mantener una conversación. Internet y el sector tecnológico no entienden de fronteras, sin inglés es complicado llegar lejos.

## **Habilidades de un desarrollador Web**

Cada vez es más frecuente encontrar desarrolladores Full Stack que dominan todas las habilidades que mencionaremos, pero también existen perfiles más focalizados que dominan menos habilidades como lo pueden ser los frontend y backend developers.

Estas son las habilidades más comunes de un desarrollador web:

- Dominio de lenguajes para la programación en backend y frontend como HTML, CSS, JavaScript.
- Saber de diseño web y cómo potenciar la Experiencia de Usuario.
- Manejo de lenguajes para administrar bases de datos como SQL.
- Conocimiento de la programación Orientada a objetos.

## **Áreas de la programación**

La programación es una carrera con mayor demanda y si eres un programador nunca te vas a quedar sin trabajo, pero, aquí viene la pregunta, ¿en qué industrias o áreas se puede trabajar como programador? En este informe les contaremos las áreas más conocidas o principales que existen, para que así puedan escoger con sabiduría el camino que desean tomar.

### **1. Desarrollo Web**

La web es todo lo que se puede ejecutar en un navegador. ¿Y qué es un navegador? Son programas como Chrome, Opera, Microsoft Edge que nos permiten acceder a sitios como Twitter, Facebook, YouTube, Slack, etc.

Podríamos dividir el desarrollo web en 2 partes:

- **Sitios web:** Son solo páginas web informativas. Su función es meramente informar al usuario, sea sobre un negocio, escuela, etc. Se puede utilizar un CMS (Sistema de Gestión de Contenidos) (Content Management System) como WordPress, Joomla, Drupal, entre otros, para crearlos.
- **Aplicaciones web:** Es una aplicación completa y contiene una lógica compleja, por ejemplo: YouTube es una aplicación web, se pueden realizar funciones como guardar vídeos, crear una transmisión, marcar favoritos, etc.

El desarrollo web se podría dividir en backend y frontend. De manera simple podemos decir que el backend es la parte que se encarga del lado del servidor (No es visible), y el frontend se encarga del lado del cliente (Lo puedes observar desde tu pantalla).

## 2. Desarrollo móvil

El desarrollo móvil en simples palabras es crear aplicaciones para teléfonos y estas pueden funcionar en 2 sistemas operativos: Android de Google y iOS de Apple.

En Android puedes desarrollar con lenguajes como Kotlin o Java, mientras que en iOS está Swift. A este tipo de aplicaciones se les conoce como "aplicaciones nativas", ya que se desarrollan específicamente en un sistema, esto quiere decir que no van a funcionar en el otro, habría que crear otra aplicación para ello. Esto puede suponer un costo excesivo para las empresas, ya que habría que hacer 2 aplicaciones.

Pero, ¿qué otras alternativas existen?

- **Aplicaciones multiplataforma:** En este tipo de aplicaciones solo se necesita desarrollar una sola vez para funcionar en Android o iOS. Se pueden hacer con React Native, Ionic, Xamarin o Flutter.

- **Progressive Web Apps:** Son aplicaciones intermedias entre web y móviles, que simulan la experiencia de una aplicación nativa. Por ejemplo, la aplicación de EDteam para móviles es una PWA.

### 3. Videojuegos

Ya todos los conocemos, tienen sus propias consolas, están en móviles, en ordenadores y consolas. Su mundo es enorme, hay diseñadores, storytelling, modelado de personajes. Entre los motores más importantes que puedes usar para desarrollar videojuegos se encuentran, Unity 3D que utiliza C# y Unreal Engine que usa C++.

### 4. Realidad virtual y aumentada

La **realidad virtual** es una inmersión total, se suele utilizar un casco o lentes que ocupa toda tu visión, y con ello puedes transportarte a un mundo de ensueño, hay proyectos muy conocidos como Beat Saber y Half-Life: Alyx.

Mientras que la realidad aumentada combina nuestra realidad con la virtual, un ejemplo de ello es Pokémon GO o los filtros de Snapchat. ¿Y cómo funciona? Es muy simple, solo se necesita un dispositivo que permita observar el entorno, y con este mismo agregarle ese añadido, por ejemplo, con la cámara del teléfono que puedes añadir efectos para las stories. Esta tecnología se puede trabajar con varios lenguajes, tales como **C#, Java, Javascript, Python**, entre otros.

Cabe mencionar que esta tecnología no está enfocada solamente en el entretenimiento, sino que se puede aplicar en áreas como la educación, medicina, mecánica, etc. Anteriormente existieron proyectos como Google Glass, que no despegó tanto como se esperaba y por su parte Microsoft sacó HoloLens.

## 5. Desarrollo de aplicaciones de escritorio

Son aplicaciones que se instalan directamente en tu sistema operativo de computadora sea Windows, Linux, Mac OS, por ejemplo: Adobe Premier, Office, un editor de código, un IDE. Para desarrollar este tipo de aplicaciones se pueden utilizar lenguajes como **Java, C#, Python**.

## 6. Sistemas Operativos / Embebidos

Los **sistemas operativos** son justamente Windows, Linux, Android o IOS, es la capa más baja de software que se comunica directamente con el hardware. Se suelen usar lenguajes como **Ensamblador** o **C** para desarrollarlos.

Mientras que los **sistemas embebidos** son programas electrónicos que realizan pocas funciones y están diseñados para cubrir necesidades específicas, casi siempre van directamente en un chip; por ejemplo: las operaciones de una lavadora, un refrigerador o algún otro electrodoméstico.

Principalmente se desarrollan con **Java** o **C**. También se relaciona con temas como Arduino, Raspberry, que se pueden programar con lenguajes como Python y Javascript.

## 7. Seguridad informática

¿Cuáles lenguajes necesitas conocer para entrar en el mundo de la seguridad informática? Principalmente Python, ya que te permite automatizar procesos, por ejemplo, podrías crear un script automático que recorra un sitio para buscar vulnerabilidades. También es bueno conocer sobre C, bash y los lenguajes en que están basados las aplicaciones que vas a auditar, asimismo SQL para las bases de datos.

## 8. Machine Learning

Básicamente consiste en enseñarle a las computadoras a través de enormes volúmenes de datos, los 2 lenguajes más importantes en el Machine Learning son **Python** y **R**.

## **9. Cloud Computing**

La nube es una red mundial de servidores que ofrecen servicios de almacenamiento, bases de datos, redes, software, análisis e inteligencia a través de internet.

### **¿Y qué lenguajes se utilizan en el Cloud Computing?**

Prácticamente todos ya que en sí la nube son servidores, pero para automatizar procesos puedes usar Python.

## **Sueldo en el Perú**

Para poder tener conocimiento de los salarios de un programador en el Perú, debemos tener en cuenta algunos puntos y factores. Dependiendo a estos factores pueden variar los salarios:

- Las funciones a realizar en la compañía o empresa.
- El rol a desempeñar. Dependiendo del rango de programador, siendo los programadores Senior uno de los mejores pagados. También están los Semisenior, Junior, entre otros.
- La empresa a contratar, ya que existes distintas cantidades de sueldos para el mismo rol en diferentes compañías. Ya que no es lo mismo ser un programador Junior en la empresa Microsoft que serlo en una empresa pequeña.
- La experiencia laboral, siendo un factor vital al momento de postular a un trabajo.
- Las metas y alcances que uno pueda tener.

De esta manera podemos entender también que va depender en que zona del Perú te encuentres, oscilando entre 1350 y 3700 soles de sueldo básico para un programador Junior (con poca experiencia), y entre 5000 y 8000 soles de sueldo básico para un programador con mayor experiencia laboral (Senior). Teniendo en cuenta que en muchas ocasiones se puede ganar más de lo mencionado.

Una de las plataformas más usadas en el Perú, “Computrabajo”, nos menciona que el sueldo medio para un programador en el Perú es de 2393 soles mensuales; dicho promedio fue sacado de una estimación de 557.098 fuentes (empresas, usuarios y empleados) en los últimos 12 meses. Siendo un promedio acertado con la información antes dicha.

Podemos destacar la mejor zona en el Perú, donde podrás ganar un mejor sueldo es en Lima, siendo una ciudad con mayor costo de vida que otras ciudades, pero siendo la ciudad con los mejores trabajos y sueldos del Perú.

## **Conclusión**

Se puede concluir que el lenguaje de programación es una parte fundamental del programa ya que, sin él, desarrollarlos ser/a muy complicado dado que facilita la manera en la que el programador plasma las ideas y deja ciertas ordenes que el lenguaje puede traducir a un lenguaje que la computadora puede entender.

Los lenguajes de programación permiten comunicarnos con la computadora, sin ellos sería imposible instruirla para que realice una acción determinada y resolver problemas tan complejos que hoy en día la sociedad exige en el comercio, la industria, el ocio, etc. A través de la historia han aparecido nuevos paradigmas y lenguajes de programación que han desarrollado su filosofía completamente o han coexistido para crear un sinnúmero de programas y aplicaciones específicos para cada tarea diferente. La programación orientada a objetos es el nuevo paradigma que ha permitido el avance de los lenguajes de programación, ya que sus conceptos se asemejan al “mundo real”, permitiendo un programa confiable, comprensible y eficiente en términos de tiempo de ejecución y consumo de espacio. El desarrollo de los lenguajes de programación va evolucionado, existiendo una gran gama de lenguajes y aplicaciones que hoy en día las escuelas deben de usarlas para el desarrollo de la enseñanza y aprendizaje de los estudiantes.

Como puede verse, existen lenguajes específicos para cada tarea diferente.

Muchos de ellos (Lenguajes de Programación) ya no se usan, mientras que las nociones de otros han sido incorporadas a otros lenguajes.

Los lenguajes y las Técnicas de Programación son importantes para la utilización de la computadora como una herramienta para resolver problemas.

En computación, un Problema consiste en la necesidad de transformar un grupo de datos iniciales en un grupo diferente de datos finales (resultados).



De este modo, una computadora podrá resolver un problema si alguien desarrolla un programa que contenga las instrucciones adecuadas que permitan transformar los datos.

Los lenguajes deben ser confiables, comprensibles, eficientes en términos de tiempo de ejecución y consumo de espacio, y deben satisfacer las necesidades de una comunidad, ya sean científicos, hombres de negocios o usuarios no técnicos. Cada uno de estos grupos está acostumbrado a un vocabulario particular y una manera de ver las cosas; de este modo, existe una gran variedad de lenguajes y muy probablemente esto continuará siendo así.