**UNIVERSIDAD**

**SALESIANA**

**DE BOLIVIA**

**CARRERA: INGENIERIA EN SOFWARE**

**COMERCIAL**



**ENSAYO**

**ESTUDIANTE: Jhael Katherine Arce Chavez**

**ASIGNATURA : Lenguaje de Programacion**

**El Impacto del Estudio de los Lenguajes de Programación en el Desarrollo de Software**

El estudio de los lenguajes de programación es esencial tanto para estudiantes de ciencias de la computación como para desabolladores profesionales. Comprender los conceptos de diseño y evaluación de lenguajes de programación va más allá de simplemente conocer uno o dos lenguajes; aumenta la capacidad para expresar ideas de manera efectiva, facilita la selección de lenguajes para proyectos específicos y mejora la eficiencia en el aprendizaje de nuevos lenguajes. Además, entender la implementan de los lenguajes ayuda a utilizarlos de manera más eficiente y a comprender mejor cómo el diseño del lenguaje afecta el desarrollo del software.

El conocimiento de los conceptos de lenguajes de programación también contribuye al avance general de la informática al fomentar la adopción de lenguajes más avanzados y mejor diseñados. Una comprensión más amplia de estos conceptos podría haber llevado a la adopción de lenguajes más sofisticados en el pasado, resaltando la importancia de que los profesionales estén bien informados al elegir los lenguajes de programación.

### **Dominios de Programación**

#### **Aplicaciones Científicas**

Los primeros lenguajes de programación de alto nivel, como Fortran, surgieron para satisfacer las necesidades de aplicaciones científicas que requerían cálculos aritméticos intensivos en coma flotante. Estos lenguajes, diseñados con eficiencia en mente debido a la competencia con el lenguaje ensamblador, se centraron en estructuras de datos como matrices y bucles de control.

#### **Aplicaciones Empresariales**

COBOL fue el primer lenguaje de alto nivel exitoso para aplicaciones empresariales, que comenzaron a desarrollarse en la década de 1950. Estos lenguajes se caracterizan por su capacidad para producir informes detallados, manipular datos decimales y de caracteres, y realizar operaciones aritméticas precisas.

#### **Inteligencia Artificial**

Lisp fue el primer lenguaje ampliamente utilizado para aplicaciones de inteligencia artificial, caracterizadas por el uso de cálculos simbólicos en lugar de numéricos. Aunque Lisp dominó este campo hasta la década de 1990, otros lenguajes, como Prolog y C, también se han utilizado en aplicaciones de IA.

#### **Software Web**

La web es compatible con una variedad de lenguajes, desde HTML para la estructura básica hasta lenguajes de programación como JavaScript y PHP para la funcionalidad dinámica. Los lenguajes de programación web han evolucionado para proporcionar interactividad y dinamismo a las aplicaciones web.

### **Diseño de Lenguajes de Programación**

#### **Tipos de Datos**

La elección de tipos de datos adecuados mejora la legibilidad y comprensión del código. Por ejemplo, el uso de un tipo booleano para indicadores clarifica la intención del código en comparación con el uso de un tipo numérico.

#### **Diseño de la Sintaxis**

El diseño sintáctico influye en la legibilidad del código. La elección de palabras y la estructura de las sentencias pueden afectar la claridad y comprensión del código, como en el caso de los grupos de sentencias difíciles de identificar.

#### **Escriturabilidad**

La facilidad para escribir programas en un lenguaje está influenciada por la simplicidad, la ortogonalidad y la expresividad del lenguaje. Un conjunto coherente de reglas y una capacidad clara para expresar ideas facilitan la escritura de programas efectivos.

#### **Confiabilidad**

La verificación de tipos y el manejo de excepciones influyen en la confiabilidad del software. La detección de errores antes de la ejecución y la capacidad de responder adecuadamente a situaciones inesperadas mejoran la confiabilidad del programa.

#### **Costos**

El costo total de un lenguaje de programación incluye el costo de entrenamiento, escritura, compilación, ejecución y mantenimiento del programa. La legibilidad y la escriturabilidad afectan los costos de mantenimiento a largo plazo.

### **Influencias en el Diseño de Lenguajes**

#### **Arquitectura de Computadoras**

La arquitectura de von Neumann ha influido en el diseño de lenguajes imperativos, aunque los lenguajes funcionales ofrecen ventajas, su adopción depende de la disponibilidad de computadoras no von Neumann eficientes.

#### **Metodologías de Diseño de Programación**

Las metodologías de diseño, como el diseño orientado a objetos, han llevado al desarrollo de nuevos lenguajes y constructos de lenguaje para respaldarlas.

### **Categorías de Lenguajes**

Los lenguajes se pueden categorizar en lenguajes imperativos, funcionales, lógicos y orientados a objetos. Los lenguajes híbridos de marcado/programación han surgido como una categoría adicional para abordar la creación de contenido web dinámico.

### **Métodos de Implementación**

#### **Compilación**

La compilación traduce programas directamente al lenguaje de máquina, lo que proporciona una ejecución rápida una vez completada la traducción.

#### **Interpretación Pura**

La interpretación pura ejecuta programas directamente desde el código fuente, lo que facilita la depuración pero puede ser considerablemente más lenta que la compilación.

#### **Sistemas de Implementación Híbridos**

Los sistemas híbridos combinan las ventajas de la compilación y la interpretación, permitiendo una ejecución rápida y una fácil depuración.

### **Entornos de Programación**

Los entornos de programación van desde sistemas básicos con un editor de texto y un compilador hasta suites integradas con múltiples herramientas accesibles a través de una interfaz gráfica. Ejemplos incluyen UNIX, Borland JBuilder, Microsoft Visual Studio .NET y NetBeans.

**Exploración Detallada del Desarrollo de Lenguajes de Programación**

En el desarrollo de los lenguajes de programación, destacando los hitos importantes y las características innovadoras que surgieron en cada período. Comenzaremos desde los primeros intentos de programación y avanzaremos hacia los lenguajes de programación modernos.

### **Plankalkül de Konrad Zuse: Un Vislumbre del Futuro**

Comenzaremos nuestro viaje con el Plankalkül de Konrad Zuse, un lenguaje de programación notablemente avanzado para su época. Aunque nunca fue implementado y su descripción no se publicó hasta décadas después, el Plankalkül sentó las bases para el diseño de lenguajes posteriores con su notación inusual y características avanzadas.

### **Pseudocódigos: Short Code y Speedcoding**

El uso de pseudocódigos como Short Code y Speedcoding marcó un hito en la historia de la programación al ofrecer soluciones más accesibles para expresar algoritmos. Short Code, desarrollado por John Mauchly en 1949, simplificó la programación al codificar expresiones matemáticas de manera legible y comprensible para los programadores. Aunque no se traducía directamente a código de máquina, su implementación con un intérprete puro facilitaba la programación en las computadoras BINAC y UNIVAC I. Por otro lado, Speedcoding, creado por John Backus para el IBM 701, permitió la realización de operaciones de punto flotante de manera eficiente, transformando la máquina en una calculadora virtual avanzada. Aunque limitado por restricciones de memoria y velocidad de ejecución, Speedcoding simplificó enormemente tareas que antes requerían semanas de trabajo.

### **El IBM 704 y Fortran: Un Paso Revolucionario**

La llegada del IBM 704 en 1954 supuso un avance significativo en la informática, ya que fue la primera computadora en incluir hardware de punto flotante. Esto eliminó la necesidad de simular operaciones de punto flotante en software, lo que permitió realizar cálculos científicos de manera más eficiente y rápida. Fortran, desarrollado por John Backus y su equipo en IBM, se diseñó para aprovechar estas capacidades del hardware. Desde su versión inicial, Fortran I, lanzada entre enero de 1955 y abril de 1957, hasta las versiones posteriores como Fortran IV, Fortran evolucionó para convertirse en uno de los lenguajes de programación más utilizados de su tiempo. Sus mejoras continuas, como la introducción de subrutinas independientes y la compatibilidad con hardware más avanzado, lo mantuvieron relevante durante décadas.

### **Functional Programming: Lisp y sus Descendientes**

Lisp, desarrollado por John McCarthy en 1958, revolucionó la programación al introducir conceptos fundamentales de la programación funcional. Diseñado para procesar listas simbólicas de manera eficiente, Lisp se destacó por su simplicidad y poder expresivo. Su modelo de datos basado en átomos y listas, junto con la aplicación de funciones a datos inmutables, promovió un estilo de programación claro y conciso. Aunque inicialmente criticado por su aparente ineficiencia, Lisp demostró ser altamente efectivo en aplicaciones de inteligencia artificial y análisis de datos. Sus dialectos derivados, como Scheme y Common Lisp, ampliaron su alcance y adaptabilidad, convirtiéndolo en un área de investigación activa en el desarrollo de lenguajes de programación.

**The First Step Toward Sophistication: ALGOL 60**

El desarrollo de ALGOL 60 marcó un hito significativo en la historia de la programación, su surgimiento estuvo motivado por la necesidad de superar las limitaciones de los lenguajes de programación específicos de la máquina, que dificultaban el intercambio de programas entre usuarios. Grupos de usuarios de computadoras en los Estados Unidos, como SHARE y USE, se unieron para solicitar a ACM que formara un comité para diseñar un lenguaje de programación científica independiente de la máquina.

La primera reunión de diseño de ALGOL 60 en Zurich en 1958 estableció los objetivos clave del nuevo lenguaje, incluida la legibilidad de los programas, su uso en publicaciones impresas y la capacidad de traducción mecánica a lenguaje de máquina

ALGOL 60 introdujo características clave como la estructura de bloque, los métodos de paso de parámetros y la capacidad de recursión. Estas innovaciones representaron diferencias significativas con respecto a ALGOL 58 y contribuyeron a la influencia y la importancia de ALGOL 60 en la historia de la programación.

Se convirtió en el estándar para comunicar algoritmos en la literatura informática durante más de dos décadas, pero también enfrentó críticas y desafíos, como la falta de compatibilidad de entrada/salida y la competencia con Fortran. Esta evaluación destaca tanto los éxitos como los fracasos de ALGOL 60, ofreciendo una visión equilibrada de su impacto en la programación.

**COBOL: La Computarización de Registros Comerciales**

El lenguaje COBOL (Common Business-Oriented Language) ha sido un pilar en el ámbito de las aplicaciones comerciales durante más de seis décadas. Aunque su influencia en el diseño de lenguajes posteriores ha sido limitada, su adaptabilidad y capacidad para satisfacer las necesidades comerciales han asegurado su longevidad.

El desarrollo de COBOL se remonta a 1959, en un momento en que la informática empresarial estaba experimentando un crecimiento similar al que había experimentado la informática científica antes del surgimiento de Fortran. FLOW-MATIC, un lenguaje compilado para aplicaciones comerciales, fue uno de los precursores de COBOL. Su diseño se basó en la necesidad de un lenguaje común que fuera fácil de usar y permitiera a los gerentes comprender los programas.

FLOW-MATIC, desarrollado por Grace Hopper en Remington-Rand UNIVAC, tuvo un impacto significativo en el diseño de COBOL. La propuesta de Hopper de escribir programas de procesamiento de datos en inglés sentó las bases para una característica clave de COBOL: su legibilidad para no programadores.

COBOL introdujo varias innovaciones, como la definición de macros y las estructuras de datos jerárquicas, que más tarde se adoptaron en otros lenguajes. Su división de datos permitió descripciones detalladas de variables y registros, haciéndolo ideal para aplicaciones de contabilidad y generación de informes.

**Los Inicios del Tiempo Compartido: BASIC**

En sus primeras etapas de desarrollo, BASIC carecía de elegancia y ofrecía un conjunto limitado de declaraciones de control.

BASIC fue diseñado con el objetivo de ser accesible para estudiantes de artes liberales. La facilidad de aprendizaje y uso, así como la capacidad de proporcionar retroalimentación rápida y privacidad, fueron consideraciones fundamentales en su diseño. La idea revolucionaria de permitir acceso compartido en tiempo real a través de terminales fue un aspecto destacado en su creación.

La primera versión de BASIC era simple y no interactiva, con solo 14 tipos de declaraciones y un solo tipo de datos: punto flotante. A pesar de sus limitaciones, su facilidad de uso lo hizo popular entre usuarios sin experiencia en ciencias.

Aunque carecía de estándares formales y su estructura de programas era deficiente, fue ampliamente utilizado en sistemas como RSTS de Digital Equipment Corporation. Su resurgimiento llegó con la aparición de Visual Basic en la década de 1990, especialmente gracias a su capacidad para crear interfaces gráficas de usuario (GUI). La versión VB.NET introdujo la programación orientada a objetos completa, agregando una nueva dimensión a la utilidad de BASIC.

**Todo para Todos: PL/I**

PL/I surgió en la década de 1960 como respuesta a las necesidades de usuarios empresariales y científicos de computadoras. IBM lideró su desarrollo como parte de la línea System/360 para reemplazar lenguajes como Fortran y COBOL.

El diseño de PL/I comenzó en 1963 con la formación de un comité compuesto por miembros de IBM y SHARE. Inicialmente concebido como una extensión de Fortran IV, evolucionó hacia un lenguaje distintivo que combinaba características de varios lenguajes existentes.

PL/I integró características innovadoras como subprogramas concurrentes, manejo de excepciones y punteros, entre otros. Fue el primer lenguaje en permitir la creación de subprogramas concurrentemente y manejar múltiples tipos de excepciones en tiempo de ejecución.

Aunque PL/I tuvo cierto éxito en aplicaciones empresariales y científicas, su complejidad y defectos de diseño limitaron su adopción generalizada. Sin embargo, su legado perdura como un hito en la evolución de los lenguajes de programación.

**Dos Primeros Lenguajes Dinámicos: APL y SNOBOL**

APL, creado por Kenneth E. Iverson en IBM, y SNOBOL, desarrollado en Bell Laboratories, son dos lenguajes de programación dinámicos destacados de la década de 1960. APL se caracteriza por su expresividad a través de un amplio conjunto de operadores simbólicos, aunque su complejidad limita su uso. En contraste, SNOBOL se enfoca en el procesamiento de texto, ofreciendo potentes funciones de coincidencia de patrones, pero con un rendimiento más lento. Ambos lenguajes han dejado una marca duradera en la historia de la informática, cada uno destacándose en áreas específicas de aplicaciones computacionales.

### **SIMULA 67 y ALGOL 68: Avances en el Diseño de Lenguajes de Programación**

### SIMULA 67, concebido por Kristen Nygaard y Ole-Johan Dahl en el Centro de Cómputo Noruego, evolucionó desde SIMULA I para abordar aplicaciones más generales, introduciendo conceptos innovadores como las corutinas y la abstracción de datos. Estas adiciones sentaron los fundamentos para la programación orientada a objetos, al extender la estructura de bloques y las declaraciones de control de ALGOL 60.

### Por otro lado, ALGOL 68, una reinvención significativa de su predecesor, ALGOL 60, se destacó por su énfasis en la ortogonalidad y la inclusión de tipos de datos definidos por el usuario. Estas características ofrecieron una flexibilidad notable en la manipulación de datos, aunque la complejidad del metalenguaje utilizado para describir ALGOL 68 y la falta de apoyo promocional limitaron su adopción.

**Algunos Descendientes Tempranos de los ALGOLs**

Los lenguajes imperativos, fundamentales en la programación, tienen sus raíces en los conceptos introducidos por ALGOL 60 y ALGOL 68. Estos influyeron de manera significativa en el diseño y la evolución de otros lenguajes.

**Simplicidad por Diseño: Pascal**

Este lenguaje, implementado en la Universidad de Stanford, introdujo innovaciones como el método de paso de parámetros por valor-resultado y la sentencia de caso, heredadas de sus predecesores. Pascal se destacó principalmente en el ámbito educativo, convirtiéndose en el lenguaje preferido para enseñar programación en muchas universidades. Sin embargo, su enfoque educativo limitó algunas características necesarias para aplicaciones más complejas, lo que condujo al surgimiento de variantes no estándar. A pesar de esto, Pascal mantuvo su popularidad durante décadas antes de ser eclipsado por lenguajes más modernos.

**Un Lenguaje de Sistemas Portátil: C**

C, concebido por Dennis Ritchie en Bell Laboratories en 1972, surgió como una evolución de lenguajes previos como BCPL y B. Influenciado por conceptos de ALGOL 68, C se destacó por su portabilidad y su implementación inicial en el sistema operativo UNIX, que lo convirtió en una herramienta fundamental para el desarrollo de sistemas.

La versatilidad de C en el control de declaraciones y la manipulación de datos lo hizo ampliamente utilizado en diversas áreas de aplicación. Sin embargo, su falta de verificación completa de tipos generó controversia, siendo considerada tanto una ventaja por su flexibilidad como una desventaja por su potencial inseguridad. La inclusión de un compilador de C en UNIX fue un factor crucial en su popularización, proporcionando a los programadores acceso gratuito a una herramienta poderosa y confiable.

**Programación Basada en la Lógica: Prolog**

La programación lógica implica el uso de una notación formal lógica para describir procesos computacionales. El cálculo de predicados es la notación principal utilizada en los lenguajes de programación lógica contemporáneos.

Los programas en lenguajes de programación lógica no especifican directamente cómo calcular un resultado, sino que describen las características del resultado deseado. El cálculo de predicados proporciona la base para comunicarse con la computadora, mientras que la resolución, desarrollada por primera vez por Robinson en 1965, ofrece la técnica de inferencia.

Alain Colmerauer, Phillippe Roussel y Robert Kowalski desarrollaron el diseño fundamental de Prolog a principios de la década de 1970. Los principales componentes de Prolog incluyen un método para especificar proposiciones de cálculo de predicados y una forma restringida de resolución. El primer intérprete de Prolog se creó en Marsella en 1972.

Los programas en Prolog consisten en colecciones de declaraciones, que pueden ser hechos o reglas. Prolog se utiliza comúnmente como una base de datos inteligente, donde los hechos y las reglas representan relaciones y pueden consultarse interactivamente.

La programación lógica ha enfrentado desafíos en términos de eficiencia y su aplicabilidad se limita a áreas específicas como sistemas de gestión de bases de datos y ciertos aspectos de la inteligencia artificial.

### **Conclusiones**

En conclusión, el estudio de los lenguajes de programación es esencial para cualquier persona interesada en la informática y el desarrollo de software. Comprender los conceptos clave relacionados con el diseño, implementación y aplicación de lenguajes de programación no solo mejora las habilidades de programación de un individuo, sino que también proporciona una base sólida para abordar los desafíos y oportunidades en constante evolución en el campo de la informática. Desde los primeros lenguajes de programación hasta los modernos, la historia de la informática ha sido moldeada por la evolución y diversificación de los lenguajes de programación, y este proceso de cambio continuo seguirá impulsando la innovación y el progreso en los años venideros.