O B J E T I V O

El presente trabajo trata de indagar en la evolución del Lenguaje C a lo largo del tiempo entre los lenguajes de programación y el avance tecnológico. Como consecuencia de la corta historia de la Informática, los lenguajes que esta utiliza tiene una existencia breve comparado con cualquier otra tecnología. Esto que podría parecer una ventaja a la hora de afrontar una revisión histórica, es un problema ya que no es frecuente encontrar estudios históricos del pasado inmediato, ya que la progresión geométrica en la aparición de los lenguajes, hace que la mayoría de los lenguajes recopilados en este trabajo sean más del presente que del pasado. El mismo pretende ubicar cronológicamente la aparición del Lenguaje C en el contexto del Hardware y las distintas generaciones de los lenguajes de programación, simplificada en las primeras paginas a modo de introducción.

H I P O T E S I S

1.− El avance tecnológico informático genera una creciente evolución

de los lenguajes de Programación provocado por el alto nivel de

necesidades de programadores y usuarios.

2.− Cuanto mayor ha sido el adelanto del hardware, mayor fue el

desarrollo del Lenguaje C.

3.− En función de los progresivos requerimientos del software se han

debido perfeccionar progresivamente las distintas versiones de C

para la realización de nuevos programas.

4.− Desde sus comienzos el Lenguaje C ha pulido su contenido, hasta la

actualidad donde posee más ventajas que inconvenientes, siendo más sus adeptos que sus detractores.

I N T R O D U C C I O N

Los ordenadores no hablan nuestro idioma, son máquinas y como tales, necesitan un lenguaje especifico pensado por el hombre para ellas. Además, necesitan constantemente interpretar todas las instrucciones que reciben. Dada la dificultad de comunicación insalvable entre el computador y el programador, pronto aparecieron lenguajes de programación que hacen posible la comunicación con el microprocesador, utilizando términos y símbolos relacionados con el tipo de problema que se debe resolver, mediante el empleo de herramientas que brinda la informática.

Estos lenguajes permiten, por un lado, escribir las operaciones que son necesarias realizar para resolver el problema de un modo parecido a como se escribiría convencionalmente (es decir, redactar adecuadamente el algoritmo de resolución del problema) y, por el otro, se encarga de traducir el algoritmo al lenguaje máquina (proceso conocido como compilación) con lo que se le confiere al programa la capacidad de correr (ser ejecutado) en el ordenador.

El ordenador es en realidad tan sólo una máquina virtual, capaz de resolver todos los problemas que los

usuarios seamos capaces de expresar mediante un algoritmo (programa).

En la actualidad hay muchos tipos de lenguajes de programación, cada uno de ellos con su propia gramática, su terminología especial y una sintaxis particular. Por ejemplo, existen algunos creados especialmente para aplicaciones científicas o matemáticas generales (BASIC, FORTRAN, PASCAL, etc.); otros, en cambio, se orientan al campo empresarial y al manejo de textos y ficheros, es decir, son en realidad fundamentalmente gestores de información (COBOL, PL/1, etc.), o muy relacionados con el lenguaje máquina del ordenador como el *C* y el ASSEMBLER.

Los ordenadores se programaban en lenguaje máquina pero las dificultades que esto conllevaba, junto con la enorme facilidad de cometer errores, cuya localización era larga y compleja, hicieron concebir, en la década de los 40, la posibilidad de usar lenguajes simbólicos. Los primeros en aparecer fueron los ensambladores, fundamentalmente consistía en dar un nombre

(mnemónico) a cada tipo de instrucción y cada dirección (etiqueta). Al principio sé hacia el programa sobre papel y , después se traducía a mano con la ayuda de unas tablas, y se introducían en la máquina en forma numérica, pero pronto aparecieron programas que se ensamblaban automáticamente.

Con el correr del tiempo y el progreso de la tecnología informática los lenguajes de programación han tenido que seguir este avance para poder satisfacer los requerimientos de los programadores y del nuevo hardware, además de las necesidades de los usuarios que crecen en un constante desarrollo.

Este trabajo consiste en hacer una reseña de cómo han evolucionado los lenguajes de programación acompañando el perfeccionamiento del hardware, haciendo especial mención al Lenguaje C.

D E S A R R O L L O

1. Definiciones de Lenguaje de Programación.

Es complicado definir qué es y qué no es un lenguaje de programación. Se asume generalmente que la traducción de las instrucciones a un código que comprende la computadora debe ser completamente sistemática. Normalmente es la computadora la que realiza la traducción. A continuación, se redactará una serie de definiciones de los lenguajes de programación :

- Un lenguaje de programación es una notación para escribir programas, a través de los cuales podemos comunicarnos con el hardware y dar así las ordenes adecuadas para la realización de un determinado proceso. Un lenguaje esta definido por una gramática o conjunto de reglas que se aplican a un alfabeto constituido por el conjunto de símbolos utilizados. Los distintos niveles de programación existentes nos permiten acceder al hardware, de tal forma que según utilicemos un nivel u otro, así tendremos que utilizar un determinado lenguaje ligado a sus correspondientes traductores.

- Conjunto de normas lingüísticas (palabras y símbolos) que permiten escribir un programa y que éste sea entendido por el ordenador y pueda ser trasladado a ordenadores similares para su funcionamiento en otros sistemas.

- Conjunto de instrucciones, ordenes y símbolos reconocibles por autómata, a través de su unidad de programación, que le permite ejecutar la secuencia de control deseada. Al conjunto de total de estas instrucciones, órdenes y símbolos que están disponibles se le suele llamar lenguajes de

programación del autómata. El programa esta formado por un conjunto de instrucciones, sentencias, bloques funcionales y grafismos que indican las operaciones a realizar.

- Las instrucciones representan la tarea más elemental de un programa: leer una entrada, realizar una operación, activar una salida, etc.
- La sentencia representa el mínimo conjunto de instrucciones o sentencias que realizan una tarea o función compleja: encontrar el valor de una función lógica en combinación de varias variables, consultar un conjunto de condiciones, etc.
- El bloque funcional es el conjunto de instrucciones o sentencias que realizan una tarea o función compleja: contadores, registros de desplazamientos, transferencias de información, etc. Todos estos elementos están relacionados entre sí mediante los símbolos o grafismos.
- Es un conjunto de palabras y símbolos que permiten al usuario generar comandos e instrucciones para que la computadora los ejecute. Los lenguajes de programación deben tener instrucciones que pertenecen a las categorías ya familiares de entrada / salida, cálculo / manipulación, de textos, lógica / comparación, y almacenamiento / recuperación.

2. Los comienzos, un poco de Historia.

Los primeros lenguajes de programación surgieron de la idea de Charles Babagge, la cual se le ocurrió a este hombre a mediados del siglo XIX. Era un profesor matemático de la universidad de Cambridge e inventor ingles, que al principio del siglo XIX predijo muchas de las teorías en que se basan los actuales

ordenadores. Consistía en lo que él denominaba la máquina analítica, pero que por motivos técnicos no pudo construirse hasta mediados del siglo XX. Con él colaboro Ada Lovedby, la cual es considerada como la primera programadora de la historia, pues realizó programas para aquélla supuesta máquina de Babagge, en tarjetas perforadas. Como la máquina no llegó nunca a construirse, los programas de Ada, lógicamente, tampoco llegaron a ejecutarse, pero si suponen un punto de partida de la programación, sobre todo si observamos que en cuanto se empezó a programar, los programadores utilizaron las técnicas diseñadas por Charles Babagge, y Ada, que consistían entre otras, en la programación mediante tarjetas perforadas. A pesar de ello, Ada ha permanecido como la primera programadora de la historia. Se dice por tanto que estos dos genios de antaño, se adelantaron un siglo a su época, lo cual describe la inteligencia de la que se hallaban dotados.

En 1823 el gobierno Británico lo apoyo para crear el proyecto de una máquina de diferencias, un dispositivo mecánico para efectuar sumas repetidas. Pero Babagge se dedicó al proyecto de la máquina analítica, abandonando la máquina de diferencias, que se pudiera programar con tarjetas perforadas, gracias a la creación de Charles Jacquard (francés). Este hombre era un fabricante de tejidos y había creado un telar que podía reproducir automáticamente patrones de tejidos, leyendo la información codificada en patrones de agujeros perforados en tarjetas de papel rígido. Entonces Babagge intento crear la máquina que se pudiera programar con tarjetas perforadas para efectuar cualquier cálculo con una precisión de 20 dígitos.

Pero la tecnología de la época no bastaba para hacer realidad sus ideas. Si bien las ideas de Babagge no llegaron a materializarse de forma definitiva, su contribución es decisiva, ya que los ordenadores actuales responden a un esquema análogo al de la máquina analítica.

En su diseño, la máquina constaba de cinco unidades básicas:

- Unidad de entrada, para introducir datos e instrucciones.
- Memoria, donde se almacenaban datos y resultados intermedios.
- Unidad de control, para regular la secuencia de ejecución de las operaciones
- Unidad Aritmético−Lógica, que efectúa las operaciones.
- Unidad de salida, encargada de comunicar al exterior los resultados.

Charles Babbage, conocido como el "padre de la informática" no pudo completar en aquella época la construcción del computador que había soñado, dado que faltaba algo fundamental: la electrónica. El camino

señalado de Babbage, no fue nunca abandonado y siguiéndolo, se construyeron los primeros computadores.

Cuando surgió el primer ordenador, el famoso ENIAC (Electronic Numerical Integrator And Calculator), su programación se basaba en componentes físicos, o sea, que se programaba, cambiando directamente el Hardware de la máquina, exactamente lo que se hacía era cambiar cables de sitio para conseguir así la programación de la máquina. La entrada y salida de datos se realizaba mediante tarjetas perforadas.

3. Clasificación de los lenguajes de programación.

3.1 Lenguaje Máquina:

El lenguaje máquina es el único que entiende directamente la computadora, ya que esta escrito en lenguajes directamente inteligibles por la máquina (computadora), utiliza el alfabeto binario, que consta de los dos únicos símbolos 0 y 1, denominados bits (abreviatura inglesa de dígitos binarios). Sus instrucciones son cadenas binarias (cadenas o series de caracteres de dígitos 0 y 1 que especifican una operación y, las posiciones (dirección) de memoria implicadas en la operación se denominan instrucciones de máquina o código máquina. Fue el primer lenguaje utilizado en la programación de computadoras, pero dejó de utilizarse por su dificultad y complicación, siendo sustituido por otros lenguajes más fáciles de aprender y utilizar, que además reducen la posibilidad de cometer errores. El lenguaje máquina es el conocido código binario. Generalmente, en la codificación de los programas se empleaba el sistema hexadecimal para simplificar el trabajo de escritura. Todas las instrucciones preparadas en cualquier lenguaje máquina tienen por lo menos dos partes. La primera es el comando u operación, que dice a las computadoras cuál es la función.

3.1.1. Ventajas del lenguaje máquina:

Posibilidad de cargar (transferir un programa a la memoria) sin necesidad de traducción posterior, lo que supone una velocidad de ejecución superior a cualquier otro lenguaje de programación.

3.1.2 Desventajas del lenguaje máquina:

Dificultad y lentitud en la codificación. Poca fiabilidad. Todas las computadoras tienen un código de operación para cada una de las funciones. La segunda parte de la instrucción es el operando, que indica a la computadora donde hallar o almacenar los datos y otras instrucciones que se van a manipular, el número de operandos de una instrucción varía en distintas computadoras.

Dificultad para verificar y poner a punto los programas. Los programas solo son ejecutables en el mismo procesador (CPU). En la actualidad, las desventajas superan a las ventajas, lo que hace prácticamente no recomendables a los lenguajes máquina.

3.2 Lenguajes de Bajo Nivel (ensamblador):

Son más fáciles de utilizar que los lenguajes máquina, pero al igual que ellos, dependen de la máquina en particular. El lenguaje de bajo nivel por excelencia es el ensamblador. El lenguaje ensamblador es el primer intento de sustituir el lenguaje máquina por otro más similar a los utilizados por las personas. Este intenta desflexibilizar la representación de los diferentes campos. Esa flexibilidad se consigue no escribiendo los campos en binario y aproximando la escritura al lenguaje. A principios de la década de los 50 y con el fin de facilitar la labor de los programadores, se desarrollaron códigos mnemotécnicos para las operaciones y direcciones simbólicas. Los códigos mnemotécnicas son los símbolos alfabéticos del lenguaje maquina. La computadora sigue utilizando el lenguaje máquina para procesar los datos, pero los programas ensambladores traducen antes los símbolos de código de operación especificados a sus equivalentes en el lenguaje maquina. En la actualidad los programadores no asignan números de dirección reales a los datos simbólicos, simplemente especifican donde quieren que se coloque la primera localidad del programa y el programa

ensamblador se encarga de lo demás, asigna localidades tanto para las instrucciones como los datos. Estos programas de ensamble o ensambladores también permiten a la computadora convertir las instrucciones en lenguaje ensamblador del programador en su propio código maquina. Un programa de instrucciones escrito en lenguaje ensamblador por un programador se llama programa fuente. Después de que el ensamblador convierte el programa fuente en código máquina a este se le denomina programa objeto. Para los programadores es más fácil escribir instrucciones en un lenguaje ensamblador que en código de lenguaje máquina pero es posible que se requieran dos corridas de computadora antes de que se puedan utilizar las instrucciones del programa fuente para producir las salidas deseadas.

El lenguaje de bajo nivel es el lenguaje de programación que el ordenador puede entender a la hora de ejecutar programas, lo que aumenta su velocidad de ejecución, pues no necesita un intérprete que traduzca cada línea de instrucciones. Visto a muy bajo nivel, los microprocesadores procesan exclusivamente señales electrónicas binarias. Dar una instrucción a un microprocesador supone en realidad enviar series de unos y ceros espaciadas en el tiempo de una forma determinada. Esta secuencia de señales se denomina código máquina. El código representa normalmente datos y números e instrucciones para manipularlos. Un modo más fácil de comprender el código máquina es dando a cada instrucción un mnemónico, como por ejemplo STORE, ADD o JUMP. Esta abstracción da como resultado el ensamblador, un lenguaje de muy bajo nivel que es específico de cada microprocesador.

Los lenguajes de bajo nivel permiten crear programas muy rápidos, pero que son, a menudo, difíciles de aprender. Más importante es el hecho de que los programas escritos en un bajo nivel sean altamente específicos de cada procesador. Si se lleva el programa a otra maquina se debe reescribir el programa desde el principio.

3.2.1. Ventajas del Lenguaje Ensamblador frente al Lenguaje Máquina:

Mayor facilidad de codificación y, en general, su velocidad de cálculo, ahorran tiempo y requieren menos atención a detalles. Se incurren en menos errores y los que se cometen son más fáciles de localizar. Tanto el lenguaje máquina como el ensamblador gozan de la ventaja de mínima ocupación de memoria y mínimo tiempo de ejecución en comparación con el resultado de la compilación del programa equivalente escrito en otros lenguajes. Los programas en lenguaje ensamblador son más fáciles de modificar que los programas en lenguaje máquina.

3.2.2. Desventajas del lenguaje ensamblador:

Dependencia total de la máquina lo que impide la transportabilidad de los programas (posibilidad de ejecutar un programa en diferentes máquinas). El lenguaje ensamblador del PC es distinto del lenguaje ensamblador del Apple Machintosh. La formación de los programadores es más compleja que la correspondiente a los programadores de alto nivel, ya que exige no solo las técnicas de programación, sino también el conocimiento del interior de la máquina. El programador ha de conocer perfectamente el hardware del equipo, ya que maneja directamente las posiciones de memoria, registros del procesador y demás elementos físicos. Todas las instrucciones son elementales, es decir, en el programa se deben describir con el máximo detalle todas las operaciones que se han de efectuar en la máquina para la realización de cualquier proceso. Los lenguajes ensamblador tienen sus aplicaciones muy reducidas, se centran básicamente en aplicaciones de tiempo real, control de procesos y de dispositivos electrónicos.

3.3. Lenguajes de alto Nivel:

Estos lenguajes son los más utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas de un modo mucho mas fácil que los lenguajes máquina y ensamblador. Un programa escrito en lenguaje de alto nivel es independiente de la máquina (las instrucciones no dependen del diseño del hardware o de una computadora en particular), por lo que estos programas son portables o

transportables. Los programas escritos en lenguaje de alto nivel pueden ser ejecutados con poca o ninguna modificación en diferentes tipos de computadoras. Son lenguajes de programación en los que las instrucciones enviadas para que el ordenador ejecute ciertas órdenes son similares al lenguaje humano. Dado que el ordenador no es capaz de reconocer estas ordenes, es necesario el uso de un intérprete que traduzca el lenguaje de alto nivel a un lenguaje de bajo nivel que el sistema pueda entender. Por lo general se piensa que los ordenadores son máquinas que realizan tareas de cálculos o procesamiento de texto. La descripción anterior es sólo una forma muy esquemática de ver una computadora. Hay un alto nivel de abstracción entre lo que se pide a la computadora y lo que realmente comprende. Existe también una relación compleja entre los lenguajes de alto nivel y el código máquina.

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, el lenguaje de alto nivel más conocido, los comandos como IF CONTADOR=10 THEN STOP pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a diez. Por desgracia para muchas personas esta forma de trabajar es un poco frustrante, dado que a pesar de que las computadoras parecen comprender un lenguaje natural, lo hacen en realidad de una forma rígida y sistemática.

Los lenguajes de alto nivel, también denominados lenguajes evolucionados, surgen con posterioridad a los anteriores (lenguaje máquina, lenguajes de bajo nivel o ensamblador) con los siguientes objetivos, entre otros:

- Lograr independencia de la máquina, pudiendo utilizar un mismo programa en diferentes equipos con la única condición de disponer de un programa traductor o compilador, que es suministrado por el fabricante, para obtener el programa ejecutable en lenguaje binario de la maquina que se trate. Además, no se necesita conocer el hardware especifico de dicha máquina. Aproximarse al lenguaje natural, para que el programa se pueda escribir y leer de una forma más sencilla, eliminando muchas de las posibilidades de cometer errores que se daban en el lenguaje máquina, ya que se utilizan palabras (en ingles) en lugar de cadenas de símbolos sin ningún significado aparente.
- Incluir rutinas de uso frecuente, como las de entrada / salida, funciones matemáticas, manejo de tablas, etc., que figuran en una especie de librería del lenguaje, de manera que se puedan utilizar siempre que se quiera sin necesidad de programarlas cada vez.

3.3.1. Ventajas de los lenguajes de alto nivel:

El tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes. La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos, nombres de las instrucciones tales como READ, WRITE, PRINT, OPEN, etc. Las modificaciones y puestas a punto de los programas son más fáciles. Reducción del costo de los programas. Transportabilidad. Permiten tener una mejor documentación. Son más fáciles de mantener.

3.3.2. Desventajas de los lenguajes de alto nivel:

Incremento del tiempo de puesta a punto al necesitarse diferentes traducciones del programa fuente para conseguir el programa definitivo. No se aprovechan los recursos internos de la máquina que se explotan mucho mejor en lenguajes máquina y ensambladores. Aumento de la ocupación de memoria. El tiempo de ejecución de los programas es mucho mayor. Se puede decir que el principal problema que presentan los lenguajes de alto nivel es la gran cantidad de ellos que existen actualmente en uso, además de las diferentes versiones o dialectos que se han desarrollado de algunos de ellos. Es difícil establecer una clasificación general de los mismos, ya que en cualquiera que se realice habrá lenguajes que pertenezcan a mas de uno de los grupos establecidos. Una clasificación muy extendida, atendiendo a la forma de trabajar de los programas y a la filosofía con que fueron concebidos, es la siguiente:

- **Lenguajes imperativos**. Utilizan instrucciones como unidad de trabajo de los programas (Cobol, Pascal, C,

Ada).

- **Lenguajes declarativos**. Los programas se construyen mediante descripciones de funciones o expresiones lógicas (Lisp, Prolog).
- **Lenguajes orientados a objetos**. El diseño de los programas se basa mas en los datos y su estructura. La unidad de proceso es el objeto y en el se incluyen los datos (variables) y las operaciones que actúan sobre ellos (Smalltalk, C++).
- **Lenguajes orientados al problema**. Diseñados para problemas específicos, principalmente de gestión, suelen ser generadores de aplicaciones.
- **Lenguajes naturales**. Están desarrollándose nuevos lenguajes con el principal objetivo de aproximar el diseño y construcción de programas al lenguaje de las personas.

4. Aparición del Lenguaje C en la Evolución del Hardware

Periodo	**Influencias**	**Lenguajes**
1950 − 55	Ordenadores primitivos	Lenguajes Ensamblador
		Lenguajes experimentales de alto
		nivel
1956 − 60	Ordenadores pequeños, caros y	FORTRAN
	Lentos	ALGOL 58 y 60
	Cintas magnéticas	COBOL
	Compiladores e interpretes	LISP
	Optimización del código	
1961 − 65	Ordenadores grandes y caros	FORTRAN IV
	Discos Magnéticos	COBOL 61 Extendido
	Sistemas operativos	ALGOL 60 Revisado
	Lenguajes De propósito general	SNOBOL
		APL (como notación sólo)
1966 − 70	Ordenadores de diferentes	PL/I
	tamaños, velocidades, costos	FORTRAN 66 (estándar)
	Sistemas de almacenamiento	COBOL 65 (estandard)
	masivo de datos (caros)	ALGOL 68
	S.O. multitarea e Interactivos	SNOBOL4
		SIMULA 67
	Compiladores con optimización	BASIC
	Lenguaje estándard , flexibles	APL/360
	y generales	
1971 − 75	Micro ordenadores	
	Sistemas de almacenamiento	PASCAL
	masivo de datos pequeños	COBOL 74
	y baratos	PL /I
	Programación Estructurada	
	Ingeniería del software	
	Lenguajes Sencillos	
1976 − 80	Ordenadores Baratos y potentes	ADA
	Sistemas distribuidos	FORTRAN 77

	Programación en Tiempo−real	PROLOG
	Programación Interactiva	**C**
	Abstracción de datos	
	Programación Con fiabilidad	
	y fácil mantenimiento	

Todo este desarrollo de las computadoras y de los lenguajes de programación, suele dividirse por generaciones y el criterio que se determinó para determinar el cambio de generación no está muy bien definido, pero resulta aparente que deben cumplirse al menos los siguientes requisitos:

- La forma en que están construidas.
- Forma en que el ser humano se comunica con ellas.

5. Generaciones de Computadoras.

- **Primera Generación.**

En esta generación había un gran desconocimiento de las capacidades de las computadoras, puesto que se realizó un estudio en esta época que determinó que con veinte computadoras se saturaría el mercado de los Estados Unidos en el campo de procesamiento de datos. Esta generación abarco la década de los cincuenta. Y se conoce como la primera generación. Estas máquinas tenían las siguientes características:

- Estas máquinas estaban construidas por medio de tubos de vacío.
- Eran programadas en lenguaje de máquina.

En esta generación las máquinas son grandes y costosas (de un costo aproximado de ciento de miles de dólares). En 1951 aparece la UNIVAC (NIVersAl Computer), fue la primera computadora comercial, que disponía de mil palabras de memoria central y podían leer cintas magnéticas, se utilizó para procesar el censo de 1950 en los Estados Unidos.

En las dos primeras generaciones, las unidades de entrada utilizaban tarjetas perforadas, retomadas por Herman Hollerith (1860 − 1929), quien además fundó una compañía que con el paso del tiempo se conocería como IBM (International Bussines Machines). Después se desarrolló por IBM la ***IBM 701*** de la cual se entregaron 18 unidades entre 1953 y 1957. Posteriormente, la compañía Remington Rand fabricó el modelo 1103, que competía con la 701 en el campo científico, por lo que la IBM desarrollo la 702, la cual presentó problemas en memoria, debido a esto no duró en el mercado.

La computadora más exitosa de la primera generación fue la IBM 650, de la cual se produjeron varios cientos. Esta computadora que usaba un esquema de memoria secundaria llamado tambor magnético, que es el antecesor de los discos actuales. Otros modelos de computadora que se pueden situar en los inicios de la segunda generación son: la UNIVAC 80 y 90, las IBM 704 y 709, Burroughs 220 y UNIVAC 1105.

- **Segunda Generación.**

Cerca de la década de 1960, las computadoras seguían evolucionando, se reducía su tamaño y crecía su capacidad de procesamiento. También en esta época se empezó a definir la forma de comunicarse con las computadoras, que recibía el nombre de programación de sistemas.

Las características de la segunda generación son las siguientes:

- Están construidas con circuitos de transistores.

- Se programan en nuevos lenguajes llamados lenguajes de alto nivel.

En esta generación las computadoras se reducen de tamaño y son de menor costo. Aparecen muchas compañías y las computadoras eran bastante avanzadas para su época como la serie 5000 de Burroughs y la ATLAS de la Universidad de Manchester. Algunas de estas computadoras se programaban con cintas perforadas y otras más por medio de cableado en un tablero. Los programas eran hechos a la medida por un equipo de expertos: analistas, diseñadores, programadores y operadores que se manejaban como una orquesta para resolver los problemas y cálculos solicitados por la administración. El usuario final de la información no tenía contacto directo con las computadoras. Esta situación en un principio se produjo en las primeras computadoras personales, pues se requería saberlas "programar" (alimentarle instrucciones) para obtener resultados; por lo tanto su uso estaba limitado a aquellos audaces pioneros que gustaran de pasar un buen número de horas escribiendo instrucciones, "corriendo" el programa resultante y verificando y corrigiendo los errores o bugs que aparecieran. Además, para no perder el "programa" resultante había que "guardarlo" (almacenarlo) en una grabadora de casete, pues en esa época no había discos flexibles y mucho menos discos duros para las PC; este procedimiento podía tomar de 10 a 45 minutos, según el programa.

El panorama se modificó totalmente con la aparición de las computadoras personales con mejores circuitos, más memoria, unidades de disco flexible y sobre todo con la aparición de programas de aplicación general en donde el usuario compra el programa y se pone a trabajar. Aparecen los programas procesadores de palabras como el célebre Word Star, la impresionante hoja de cálculo (spreadsheet) Visicalc y otros más que de la noche a la mañana cambian la imagen de la PC. El software empieza a tratar de alcanzar el paso del hardware. Pero aquí aparece un nuevo elemento: el usuario.

El usuario de las computadoras va cambiando y evolucionando con el tiempo. De estar totalmente desconectado a ellas en las máquinas grandes pasa la PC a ser pieza clave en el diseño tanto del hardware como del software. Aparece el concepto de Human Interfase que es la relación entre el usuario y su computadora. Se habla entonces de hardware ergonómico (adaptado a las dimensiones humanas para reducir el cansancio), diseños de pantallas antirreflejos y teclados que descansen la muñeca. Con respecto al software se inicia una verdadera carrera para encontrar la manera en que el usuario pase menos tiempo capacitándose y entrenándose y más tiempo produciendo. Se ponen al alcance programas con menús (listas de opciones) que orientan en todo momento al usuario (con el consiguiente aburrimiento de los usuarios expertos); otros programas ofrecen toda una artillería de teclas de control y teclas de funciones (atajos) para efectuar toda suerte de efectos en el trabajo (con la consiguiente desorientación de los usuarios novatos). Se ofrecen un sinnúmero de cursos prometiendo que en pocas semanas hacen de cualquier persona un experto en los programas comerciales. Pero el problema "constante" es que ninguna solución para el uso de los programas es "constante". Cada nuevo programa requiere aprender nuevos controles, nuevos trucos, nuevos menús. Se empieza a sentir que la relación usuario−PC no está acorde con los desarrollos del equipo y de la potencia de los programas. Hace falta una relación amistosa entre el usuario y la PC.

Las computadoras de esta generación fueron: la Philco 212 (esta compañía se retiró del mercado en 1964) y la UNIVAC M460, la Control Data Corporation modelo 1604, seguida por la serie 3000, la IBM mejoró la 709 y sacó al mercado la 7090, la National Cash Register empezó a producir máquinas para proceso de datos de tipo comercial, introdujo el modelo NCR 315. La Radio Corporation of America introdujo el modelo 501, que manejaba el lenguaje COBOL, para procesos administrativos y comerciales. Después salió al mercado la RCA 601.

- **Tercera generación**

Con los progresos de la electrónica y los avances de comunicación con las computadoras en la década de los 1960, surge la ***tercera generación*** de las computadoras. Se inaugura con la IBM 360 en abril de 1964. Las características de esta generación fueron las siguientes:

- Su fabricación electrónica esta basada en circuitos integrados.
- Su manejo es por medio de los lenguajes de control de los sistemas operativos.

La IBM produce la serie 360 con los modelos 20, 22, 30, 40, 50, 65, 67, 75, 85, 90, 195 que utilizaban técnicas especiales del procesador, unidades de cinta de nueve canales, paquetes de discos magnéticos y otras características que ahora son estándares (no todos los modelos usaban estas técnicas, sino que estaba dividido por aplicaciones). El sistema operativo de la serie 360, se llamó OS que contaba con varias configuraciones, incluía un conjunto de técnicas de manejo de memoria y del procesador que pronto se convirtieron en estándares. En 1964 CDC introdujo la serie 6000 con la computadora 6600 que se consideró durante algunos años como la más rápida.

En la década de 1970, la IBM produce la serie 370 (modelos 115, 125, 135, 145, 158, 168). UNIVAC compite son los modelos 1108 y 1110, máquinas en gran escala; mientras que CDC produce su serie 7000 con el modelo 7600. Estas computadoras se caracterizan por ser muy potentes y veloces.

A finales de esta década la IBM de su serie 370 produce los modelos 3031, 3033, 4341. Burroughs con su serie 6000 produce los modelos 6500 y 6700 de avanzado diseño, que se reemplazaron por su serie 7000. Honey − Well participa con su computadora DPS con varios modelos.

A mediados de la década de 1970, aparecen en el mercado las computadoras de tamaño mediano, o **minicomputadoras** que no son tan costosas como las grandes (llamadas también como **mainframes** que significa también, gran sistema), pero disponen de gran capacidad de procesamiento. Algunas minicomputadoras fueron las siguientes: la PDP − 8 y la PDP − 11 de Digital Equipment Corporation, la VAX (Virtual Address eXtended) de la misma compañía, los modelos NOVA y ECLIPSE de Data General, la serie 3000 y 9000 de Hewlett − Packard con varios modelos el 36 y el 34, la Wang y Honey − Well −Bull, Siemens de origen alemán, la ICL fabricada en Inglaterra. En la Unión Soviética se utilizó la US (Sistema Unificado, Ryad) que ha pasado por varias generaciones.

- **Cuarta Generación**

Aquí aparecen los **microprocesadores** que es un gran adelanto de la microelectrónica, son circuitos integrados de alta densidad y con una velocidad impresionante. Las microcomputadoras con base en estos circuitos son extremadamente pequeñas y baratas, por lo que su uso se extiende al mercado industrial. Aquí nacen las computadoras personales que han adquirido proporciones enormes y que han influido en la sociedad en general sobre la llamada **"revolución informática".**

En 1976 Steve Wozniak y Steve Jobs inventan la primera **microcomputadora** de uso masivo y más tarde forman la compañía conocida como la Apple que fue la segunda compañía más grande del mundo, antecedida tan solo por IBM; y esta por su parte es aún de las cinco compañías más grandes del mundo. En 1981 se vendieron 80000 computadoras personales, al siguiente subió a 1.400.000. Entre 1984 y 1987 se vendieron alrededor de 60 millones de computadoras personales, por lo que no queda duda que su impacto y penetración han sido enormes.

Con el surgimiento de las computadoras personales, el software y los sistemas que con ellas de manejan han tenido un considerable avance, porque han hecho más interactiva la comunicación con el usuario. Surgen otras aplicaciones como los procesadores de palabra, las hojas electrónicas de cálculo, paquetes gráficos, etc. También las industrias del Software de las computadoras personales crecen con gran rapidez, Gary Kildall y William Gates se dedicaron durante años a la creación de sistemas operativos y métodos para lograr una utilización sencilla de las microcomputadoras (son los creadores de CP/M y de los productos de Microsoft).

No todo son microcomputadoras, por su puesto, las minicomputadoras y los grandes sistemas continúan en desarrollo. De hecho las máquinas pequeñas rebasaban por mucho la capacidad de los grandes sistemas de 10

o 15 años antes, que requerían de instalaciones costosas y especiales, pero sería equivocado suponer que las grandes computadoras han desaparecido; por el contrario, su presencia era ya ineludible en prácticamente todas las esferas de control gubernamental, militar y de la gran industria. Las enormes computadoras de las series CDC, CRAY, Hitachi o IBM por ejemplo, eran capaces de atender a varios cientos de millones de operaciones por segundo.

Con la popularidad de las microcomputadoras muchas compañías comenzaron a implementar su propio C por lo cual surgieron discrepancias entre sí.

Por esta razón ANSI (American National Standars Institute, por sus siglas en inglés) , estableció un comité en 1983 para crear una definición no ambigua del lenguaje C e independiente de la máquina que pudiera utilizarse en todos los tipos de C.

- **Quinta Generación**

En vista de la acelerada marcha de la microelectrónica, la sociedad industrial se ha dado a la tarea de poner también a esa altura el desarrollo del software y los sistemas con que se manejan las computadoras. Surge la competencia internacional por el dominio del mercado de la computación, en la que se perfilan dos líderes que, sin embargo, no han podido alcanzar el nivel que se desea: la capacidad de comunicarse con la computadora en un lenguaje más cotidiano y no a través de códigos o lenguajes de control especializados.

Japón lanzó en 1983 el llamado "programa de la quinta generación de computadoras", con los objetivos explícitos de producir máquinas con innovaciones reales en los criterios mencionados. Y en los Estados Unidos ya está en actividad un programa en desarrollo que persigue objetivos semejantes, que pueden resumirse de la siguiente manera:

- Procesamiento en paralelo mediante arquitecturas y diseños especiales y circuitos de gran velocidad.
- Manejo de lenguaje natural y sistemas de inteligencia artificial.

El futuro previsible de la computación es muy interesante, y se puede esperar que esta ciencia siga siendo objeto de atención prioritaria de gobiernos y de la sociedad en conjunto.

6. Aparición del Lenguaje C dentro de las Generaciones de los

Lenguajes de Programación

Las generaciones de los lenguajes de programación, se han venido dando debido a que las necesidades que plantean los problemas son cada día más grandes y complejos, a continuación se hace un pequeño resumen de cada una de las generaciones de lenguajes de programación :

- **Primera Generación:** Los lenguajes de primera generación o también conocidos como lenguajes máquina, son en los que se utiliza el código binario (unos y ceros) para comunicarse con la computadora, esta generación de lenguajes es muy complicada, ya que al usar pocos signos, no puede expresar cosas muy complicadas. En la actualidad ya casi no se trabaja con lenguajes máquina, los únicos que lo hacen son los diseñadores de los "chips" de los procesadores.
- **Segunda Generación:** Los lenguajes de esta segunda generación son conocidos también como ensambladores, y se distinguen de los lenguajes máquina por su eficiencia (en comparación con sus antecesores). Estos lenguajes ensambladores se basan en lo que es la comprensión de varias palabras en una sola, por ejemplo:

ADC significara "sumar con reserva" (en ingles: ADd with Carry).

Haciendo notoria la aclaración, de que esta serie de instrucciones serán traducidas al lenguaje máquina por el compilador del lenguaje.

- **Tercera Generación:** Los lenguajes de tercera generación o de alto nivel son los lenguajes más comunes o que más conocemos, **el Lenguaje C se desarrolla dentro de esta generación,** además podemos mencionar a Pascal, Algol, Cobol, Fortran, BASIC. Estos lenguajes se asemejan ya un poco más al lenguaje humano, al utilizar palabras completas (en ingles) para la codificación de los programas.
- **Cuarta Generación**: Son los lenguajes de "programación asistida" por medio de ayudantes o wizards, estos lenguajes se han diseñado para facilitar la realización de muy variadas tareas, como lo son la simulación de fenómenos físicos, manipulación de datos estadísticos, etc. Algunos de estos lenguajes son: Visual Basic, INFORMIX 4GL, Visual J++, Visual C, he inclusive algunos autores consideran las planillas de calculo dentro de esta generación.
- **Quinta Generación:** En esta generación, el programador solo ingresa hechos y hace consultas, no se preocupa de cómo hacer los algoritmos que entregan la respuesta, algunos autores hasta hace poco todavía consideraban a esta generación como un sueño, pero gracias al avance de la tecnología, hoy en día es toda una realidad, como lo veremos a continuación.

Ahora que ya hemos hecho un breve análisis de los lenguajes de programación, continuaremos con el análisis del Lenguaje C.

7. Marco Histórico del Lenguaje C.

A finales de los 60 principios de los 70, Ken Thompson, un programador de sistemas de los Laboratorios Bell, desarrolla el lenguaje B, el cual sería el precursor del C, con intención de codificar sus programas y algoritmos, para probar y experimentar con estructuras, servicios y teorías de eficiencia, que posteriormente Brian Kernighan bautizaría con el nombre de Unix, que en la fase de arranque esta escrito en ensamblador, en vistas a su transportabilidad a otras máquinas. B era un lenguaje evolucionado e independiente de la máquina, inspirado en el lenguajes BCPL concedido por Martin Richard en 1967. No se imaginaron que con reestructurar y agregar instrucciones al Lenguaje B, añadirle estructuras de datos y tipos, pasaje de parámetros a funciones recursivas, apuntadores a funciones y unas sencillas modificaciones, Dennis Ritchie también de Laboratorios Bell, de AT&T en 1972 crearía el primer Lenguaje C para las primeras maquinas DEC PDP−11.

AT&T lo enserio como un compilador (comp. C) llamado K&R C que junto con el sistema operativo UNIX empezaron a invadir universidades. Después, cada persona que adquiría una copia de UNIX recibía un compilador de C gratis. El lenguaje mas popular fue C. Por lo tanto UNIX fue escrito en C. Entonces si se quería entender UNIX se tenia que aprender C. La característica era que C era gratis y entonces nadie se sentía presionado a aprenderlo. Cual fue el resultado ? Un gran estándar.

Luego C se convirtió en un gran estándar, entonces las compañías introducían sus propios compiladores C. Incluyendo que pudiesen ejecutarse en otros sistemas operativos que no fuesen UNIX. Cada uno de estos compiladores introducía ensanchamientos diseñados para mejorar las limitaciones que mostraba el modelo original. Pero las modificaciones que cada quien hacia traía como resultado la incompatibilidad de las versiones entre si, entonces incrementaba la demanda por un estándar a nivel nacional. Entonces en 1987 nació el primer estándar "The American National Standars Institute (ANSI) versión of C " esta versión fue mejor conocida como ANSI C o C estándar. C++ esta basado en estos compiladores y por lo tanto es el mas compatible con ANSI C.

Habitualmente se citan cuatro lenguajes que se entienden fueron los antepasados del C actual: Algol68, CPL, BCPL y B.

No nos detendremos en su estudio mas profundo porque no es objeto de este documento.

La novedad que proporcionó el lenguaje C sobre el B fue el diseño de tipos y estructuras de datos. Los tipos básicos de datos eran char (carácter), int (entero), float (reales en simple precisión) y double (reales en doble precisión). Posteriormente se añadieron los tipos short (enteros de longitud menor a la del int), long (enteros de longitud mayor a la del int) y enumeraciones. Los tipos estructurados básicos de C son las estructuras, las uniones y los arrays. Estos permiten la de definición y declaración de tipos derivados de mayor complejidad.

Las instrucciones de control de flujo de C son las habituales de la programación estructurada: *if*, *for*, *while*, *swith−case*, todas incluidas en su predecesor BCPL.

Desde su nacimiento se fue implantando como el lenguaje de programación de sistemas favorito para muchos programadores, sobre todo por ser un lenguaje que conjugaba la abstracción de los lenguajes de alto nivel con la eficiencia del lenguaje máquina. Los programadores de sistemas que trabajaban sobre MS−DOS y Macintosh también utilizaban C, con lo cual la práctica totalidad de aplicaciones de sistema para microordenadores y para sistemas UNIX está escrita en este lenguaje.

A mediados de los ochenta el C se convierte en un estándar internacional ISO. Este estándar incluye tanto la definición del lenguaje como una enorme biblioteca de funciones para entrada/salida, tratamiento de textos, matemáticas, etc. A mediados de los ochenta se crea el C++, extensión de C orientada a objetos. El C++ se convierte en estándar ISO en 1998. En el momento actual, el lenguaje C no va a modificarse más. Será el C++ el que incorporará nuevos cambios.

Originariamente, el manual de referencia del lenguaje para el gran público fue el libro de Kernighan y Ritchie, escrito en 1977. Es un libro que explica y justifica totalmente el desarrollo de aplicaciones en C, aunque en él se utilizaban construcciones, en la definición de funciones, que podían provocar confusión y errores de programación que no eran detectados por el compilador. Como los tiempos cambian y las necesidades también, en 1983 ANSI establece el comité X3J11 para que desarrolle una definición moderna y comprensible del C. El estándar está basado en el manual de referencia original de 1972 y se desarrolla con el mismo espíritu de sus creadores originales. La primera versión de estándar se publicó en 1988 y actualmente todos los compiladores utilizan la nueva definición. Una aportación muy importante de ANSI consiste en la definición de un conjunto de librerías que acompañan al compilador y de las funciones contenidas en ellas. Muchas de las operaciones comunes con el sistema operativo se realizan a través de estas funciones. Una colección de ficheros de encabezamiento, headers, en los que se definen los tipos de datos y funciones incluidas en cada librería. Los programas que utilizan estas bibliotecas para interactuar con el sistema operativo obtendrán un comportamiento equivalente en otro sistema.

Actualmente, C es el Lenguaje de programación preferido para el desarrollo de Herramientas, Editores, Manejadores de Bases de Datos, Compiladores e Interpretadores y Traductores de Lenguaje, Generadores de Sistemas Expertos, Sistemas Operativos, Procesadores de Palabras, Paquetes de Comunicación y Teleproceso, Hojas de Cálculos, Aplicaciones de CAD/CAM, y toda una infinidad de productos.

7.1. Aparición de C++ como Lenguaje Orientado a Objetos.

En la década de 1970 se volvió popular el concepto de objeto entre los investigadores de los lenguajes de programación. Un objeto es un conjunto de códigos, datos diseñados para emular o imitar una entidad física o abstracta. Los objetos son eficientes como elementos de programación por dos razones principales: representan una abstracción directa de los elementos que se utilizan comúnmente y ocultan la mayor parte de la complejidad de su implantación a los usuarios. Los primeros objetos que se desarrollaron fueron aquellos que estaban más íntimamente ligados a las computadoras, como INTERGER, ARRAY y STACK. Además se diseñaron lenguajes como el SmallTalk el cual es ya ortodoxo, donde se definía todo como un objeto.

8. Características del Lenguaje C

El lenguaje C se conoce como un lenguaje compilado. Existen dos tipos de lenguaje: interpretados y compilados. Los interpretados son aquellos que necesitan del código fuente para funcionar (Basic). Los compilados convierten el código fuente en un fichero objeto y éste en un fichero ejecutable. Este es el caso del lenguaje C . Podemos decir que el lenguaje C es un lenguaje de nivel medio, pero muy versátil y eficiente, que revolucionó las técnicas y estilo de programación, ya que combina elementos de lenguaje de alto nivel con la funcionalidad del lenguaje ensamblador.

Se caracteriza por ser un lenguaje estructurado, es decir, el programa se divide en módulos (funciones) independientes entre sí, que permite crear procedimientos en bloques dentro de otros procedimientos.

Sigue el paradigma de la programación estructurada:

Algoritmos + estructuras de datos = programas

El lenguaje C inicialmente fue creado para la programación de:

- Sistemas operativos
- Intérpretes
- Editores
- Ensambladores
- Compiladores
- Administradores de bases de datos.

Actualmente, debido a sus características, puede ser utilizado para todo tipo de programas.

Hay que destacar principalmente que el C es un lenguaje portable, que puede utilizar el mismo código en diferentes equipos y sistemas informáticos: el lenguaje es independiente de la arquitectura de cualquier maquina en particular y del sistema operativo que se utiliza para desarrollar aplicaciones portables.

Desarrollado originalmente para implementar el Sistema Operativo Unix y sus Herramientas, C provee las misma facilidades para la manipulación de bytes de un lenguaje assembler combinadas con instrucciones estructuradas de control de flujo condicionado y manipulación de tipos y estructuras de Datos de los lenguajes de tercera generación.

C es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. Por ser un lenguaje de nivel intermedio es sencillo y no está especializado en ningún tipo de aplicación. Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y sobre todo, se aprende rápidamente. En poco tiempo, un programador puede utilizar la totalidad del lenguaje.

Este lenguaje ha sido estrechamente ligado al sistema operativo UNIX, puesto que fueron desarrollados conjuntamente. Se le suele llamar lenguaje de programación de sistemas debido a su utilidad para escribir compiladores y sistemas operativos, aunque de igual forma se pueden desarrollar cualquier tipo de aplicación.

La base del C proviene del BCPL, escrito por Martín Richards, y del B escrito por Ken Thompson en 1970 para el primer sistema UNIX en un DEC PDP−7. Estos son lenguajes sin tipos, al contrario que el C que proporciona varios tipos de datos. Los tipos son caracteres, números enteros y en coma flotante, de varios tamaños. Además se pueden crear tipos derivados mediante la utilización de punteros, vectores, registros y uniones.

C trabaja con tipos de datos que son directamente tratables por el hardware de la mayoría de computadoras actuales, como son los caracteres, números y direcciones. Estos tipos de datos pueden ser manipulados por las

operaciones aritméticas que proporcionan las computadoras. No proporciona mecanismos para tratar tipos de datos que no sean los básicos, debiendo ser el programador el que los desarrolle. Esto permite que el código generado sea muy eficiente y de ahí el éxito que ha tenido como lenguaje de desarrollo de sistemas. No proporciona otros mecanismos de almacenamiento de datos que no sea el estático y no proporciona mecanismos de entrada ni salida. Ello permite que el lenguaje sea reducido y los compiladores de fácil implementación en distintos sistemas. Por el contrario, estas carencias se compensan mediante la inclusión de funciones de librería para realizar todas estas tareas, que normalmente dependen del sistema operativo.

Las instrucciones de control de flujo de C son las habituales de la programación estructurada: IF, FOR, WHILE, SWITCH − CASE, todas incluidas en su predecesor BCPL.

C incluye también punteros y funciones. Los argumentos de las funciones se pasan por valor, esto es copiando su valor, lo cual hace que no se modifiquen los valores de los argumentos en la llamada. Cuando se desea modificar los argumentos en la llamada, éstos se pasan por referencia, es decir, se pasan las direcciones de los argumentos. Por otra parte, cualquier función puede ser llamada recursivamente.

Una de las peculiaridades de C es su riqueza de operadores. Puede decirse que prácticamente dispone de un operador para cada una de las posibles operaciones en código máquina. Hay toda una serie de operaciones que pueden hacerse con el lenguaje C, que realmente no están incluidas en el compilador propiamente dicho, sino que las realiza un preprocesador justo antes de cada compilación. Las dos más importantes son #define (directriz de sustitución simbólica o de definición) e #include (Directriz de inclusión en el fichero fuente).

Finalmente, C, que ha sido pensado para ser altamente transportable y para programar lo improgramable, igual que otros lenguajes tiene sus inconvenientes:

- Carece de instrucciones de entrada/salida, de instrucciones para manejo de cadenas de caracteres, con lo que este trabajo queda para la librería de rutinas, con la consiguiente pérdida de transportabilidad.
- La excesiva libertad en la escritura de los programas puede llevar a errores en la programación que, por ser correctos sintácticamente no se detectan a simple vista.
- Por otra parte las precedencias de los operadores convierten a veces las expresiones en pequeños rompecabezas.

A pesar de todo, C ha demostrado ser un lenguaje extremadamente eficaz y expresivo.

Este lenguaje ha evolucionado paralelamente a UNIX, que a su vez ha pasado por diversas versiones entre las que destaca la de Microsoft con su XENIT para micros de 16 bits.

Algunos de las C existentes son:

- Quick C
- C++
- Turbo C
- Turbo C ++
- Borland C
- Borland C++
- Microsoft C

9. Componentes del Lenguaje C .

Sigue el paradigma de la programación estructurada:

Algoritmos + estructuras de datos = programas.

9.1. Estructuras de datos.

- Literales
- Tipos básicos (todos numéricos)
- Tipos enumerados
- Tipos estructurados (struct, union)
- Punteros y vectores

9.2. Construcciones algorítmicas.

- Construcciones condicionales (if,switch).
- Construcciones iterativas(while,for,do...while).
- Subrutinas (funciones) .

- **Otros Elementos de C :**

- Comentarios .
- Inclusión de ficheros .
- Macros .
- Compilación condicional .

El preprocesador es quien normalmente se encarga de interpretar estas construcciones.

9.4. Características destacadas.

- Orientado a la programación de sistemas .
- Es altamente transportable .
- Es muy flexible .
- Genera código muy eficiente.
- Es muy expresivo (se pueden realizar muchas funciones escribiendo pocas líneas de código) .
- Es muy poco modular .
- Hace pocas comprobaciones .
- Da poca disciplina al programador .
- Es difícil leer código escrito por otras personas .

10. Elementos generales de un programa en C .

Aunque cada uno de los programas son distintos, todos tienen características comunes. La estructura y los elementos de un programa en C son los siguientes:

Comentarios

Inclusión de archivos

main()

{

variables locales

flujo de sentencias

}

Definición de funciones creadas por el programador utilizadas en main()

10.1. Veamos en que consiste cada uno:

Comentarios:

Se identifican porque van entre diagonales y asterisco. Nos sirve para escribir información que nos referencie al programa pero que no forme parte de él. Por ejemplo especificar que hace el programa, quien lo elaboró, en que fecha, que versión es, etc.

Inclusión de archivos:

Consiste en mandar llamar a la o las bibliotecas donde se encuentran definidas las funciones de C (instrucciones) que estamos utilizando en el programa.

En realidad, la inclusión de archivos no forma parte de la estructura propia de un programa sino que pertenece al desarrollo integrado de C. Se incluye aquí para que no olvidar que se debe mandar llamar a los archivos donde se encuentran definidas las funciones estándar que va a utilizar.

main():

En C, todo está constituido a base de funciones. El programa principal no es la excepción. main() indica el comienzo de la función principal del programa la cual se delimita con llaves.

Variables locales:

Antes de realizar alguna operación en el programa, se deben declarar la(s) variable(s) que se utilizarán en el programa.

Flujo de sentencias:

Es la declaración de todas las instrucciones que conforman el programa.

Definición de funciones creadas por el programador utilizadas en main(): Finalmente, se procede a definir el contenido de las funciones utilizadas dentro de main(). Estas contienen los mismos elementos que la función principal.

C O N C L U S I O N E S

El C tiene sus detractores, como no podía ser de otra manera, sin embargo lo que nadie puede negar es que es uno de los lenguajes de programación más extendidos (hay quién afirma que el que más). Además, la historia del C está tremendamente ligada a la de los sistemas operativos UNIX y , por tanto, la mayor parte del software desarrollado para dichos sistemas está hecho en C.

Críticas del Lenguaje C

Dentro de las principales críticas que le asechan podemos destacar :

- Lo poco estricto que es el lenguaje con la comprobación de los tipos de datos, dejando esta tarea muchas veces en manos del programador.

- El no verificar automáticamente los límites de los vectores.
- La repetición que hace de símbolos en operadores diferentes (=,*,−). (Sobrecarga de operadores).
- El no poder anidar funciones, con lo que se dificulta la estructuración y la abstracción de datos.
- La incertidumbre existente en el orden de evaluación de las listas de expresiones y parámetros.
- La facilidad con que los programas pueden derivar hacia ser crípticos.

Tampoco es prudente negar la evolución que ha experimentado a lo largo de los primeros pasos que ha dado con el nombre de B, hasta hoy en donde podemos observar que se ha adaptado con holgura a los avances de hardware como de software. Incluso en el desarrollo de C++, podemos apreciar que también se ha adecuado a las necesidades que ha surgido en la programación orientada a objetos. En el campo de los lenguajes estructurados se destacado como uno de los mas utilizados y que siguen estando en vigencia aún en la actualidad tanto en el ámbito laboral, como educativo.

Bondades del Lenguaje C.

Así como críticas también posee ventajas que hacen que sus adeptos lo consideren de la forma siguiente:

- Como un lenguaje de programación estructurado de mediano nivel, pero muy versátil y eficiente, que revolucionó las técnicas y estilo de programación.
- C es un lenguaje independiente del sistema operativo que se utiliza para desarrollar aplicaciones portables.
- C provee las misma facilidades para la manipulación de bytes de un lenguaje assembler combinadas con instrucciones estructuradas de control de flujo condicionado y manipulación de tipos y estructuras de Datos de los lenguajes de tercera generación.
- En Lenguaje C, lo fácil es simple y lo difícil posible.
- Un buen programador de C, utiliza librerías y desarrolla programas modulares con funciones bien estructuradas, que son fáciles de mantener.

B I B L I O G R A F I A

Los documentos empleados para la realización de este trabajo han sido obtenidos de:

- Delphi 5. Autor: Francisco Charte. Editorial ANAYA Multimedia 1999. (681.3 CHA del).
- Fundamentos de Informática. Autor: Luis A. Ureña y cols. Editorial ra−ma 1997 (681.3 FUN).
- Java 2 EDICCION 2000. Autor Miguel Angel Martín Tardio. ANAYA Multimedia 2000. (681.3 MAR man).

- VBScript y programación ASP. Autor: Oscar González Moreno. Editorial ANAYA Multimedia 1997. (681.3 GON vbs)

- http://www.puc.cl/curso_dist/cbc/textos/teoria/lengua2.html
- http://lawebdelprogramador.com/
- www.tsi.com.mx/das/roman/manual/u1.htm#a1_3_4
- http://www.geocities.com/SiliconValley/2915/manual.htm
- http://html.programacion.net/xhtml/capitulo1.htm
- www.geocities.com
- www.lycos.es
- www.google.com
- www.programando.com
- www.terra.es
- www.ya.com

1


UNIVERSIDAD ARGENTINA
TRINUS ET UNUS
uk.
VITA SPIRITUS ET SOCIETAS
JOHN F. KENNEDY