

# LENGUAJES DE ANTO NIVEL

---

PYTHON

BASIC

ALGOL



ADA

JAVA

# LENGUAJES DE ALTO NIVEL

<b>1) Introduccion</b> .....	3
<b>2) Ada</b> .....	3
A) <i>Definicion:</i> .....	3
B) <i>Historia:</i> .....	4
C) <i>Caracteristicas</i> .....	4
<b>3) Algol:</b> .....	5
A) <i>Ejemplo:</i> .....	5
B) <i>Algol w:</i> .....	5
C) <i>Algol 68:</i> .....	6
<b>4) Basic:</b> .....	6
A) <i>Ejemplo:</i> .....	6
B) <i>Nacimiento y primeros años:</i> .....	7
c) <i>Visual basic:</i> .....	8
<b>5) Java</b> .....	9
A) <i>Historia reciente</i> .....	9
C) <i>Ejemplos</i> .....	11
<b>6) Python</b> .....	11
A) <i>Introduccion</i> .....	11
B) <i>Ejemplos:</i> .....	12
<b>7) Pascal</b> .....	13
A) <i>Caracteristicas</i> .....	13
B) <i>Ejemplo:</i> .....	14
C) <i>Compiladores disponibles:</i> .....	14
<b>8) C++</b> .....	15
A) <i>Introduccion</i> .....	15
B) <i>Principios</i> .....	15
C) <i>Concepto de clase</i> .....	16
<b>9) C (lenguaje de nivel medio)</b> .....	18
A) <i>Introduccion</i> .....	18
B) <i>Ventajas</i> .....	18
C) <i>Variantes</i> .....	18
D) <i>Ejemplo</i> .....	19
<b>10) C#</b> .....	19
A) <i>Concepto</i> .....	19

# **1) INTRODUCCION**

La programación en un lenguaje de bajo nivel como el lenguaje de la máquina o el lenguaje simbólico tiene ciertas ventajas:

- Mayor adaptación al equipo.
- Posibilidad de obtener la máxima velocidad con mínimo uso de memoria.

Pero también tiene importantes inconvenientes:

- Imposibilidad de escribir código independiente de la máquina.
- Mayor dificultad en la programación y en la comprensión de los programas.

Por esta razón, a finales de los años 1950 surgió un nuevo tipo de lenguaje que evitaba los inconvenientes, a costa de ceder un poco en las ventajas.

Estos lenguajes se llaman "de tercera generación" o "de alto nivel", en contraposición a los "de bajo nivel" o "de nivel próximo a la máquina".

Los lenguajes de Alto Nivel se caracterizan por el uso de macroinstrucciones, las cuales en el momento de ser ejecutadas por el computador tienen que ser traducidas a lenguaje de máquina, ya sea por medio de un compilador o intérprete. En los primeros lenguajes de alto nivel la limitante era que se orientaban a un área específica y sus instrucciones requerían de una sintaxis predefinida, se clasifican como lenguajes procedimentales.

Otra limitante de los lenguajes de alto nivel es que se requieren de ciertos conocimientos de programación para realizar las secuencias de instrucciones lógicas.

Para que el común de los usuarios pudiesen solucionar su problemática de procesamiento de datos en una forma más rápida y fácil se crean los lenguajes de muy alto nivel.

A continuación se explican cada uno de los lenguajes de alto nivel

## **2) ADA**

### *A) DEFINICION:*

Ada es un lenguaje de programación estructurado y fuertemente tipado de forma estática que fue diseñado por Jean Ichbiah de CII Honeywell Bull por encargo del Departamento de Defensa de los Estados Unidos (DoD). Es un lenguaje multipropósito, orientado a objetos y concurrente, pudiendo llegar desde la facilidad de Pascal hasta la flexibilidad de C++.

Fue diseñado con la seguridad en mente y con una filosofía orientada a la reducción de errores comunes y difíciles de descubrir. Para ello se basa en un tipado muy fuerte y en chequeos en tiempo de ejecución (desactivables en beneficio del rendimiento). La sincronización de tareas se realiza mediante la primitiva rendezvous.

Ada se usa principalmente en entornos en los que se necesita una gran seguridad y confiabilidad como la defensa, la aeronáutica (Boeing o Airbus), la gestión del tráfico aéreo (como Indra en España) y la industria aeroespacial entre otros

## *B) HISTORIA:*

El lenguaje fue diseñado bajo encargo del DoD. Durante los años 1970, este departamento tenía proyectos en una infinidad de lenguajes y estaba gastando mucho dinero en software. Para solucionarlo se buscó un lenguaje único que cumpliera unas ciertas normas recogidas en el documento Steelman. Después de un estudio de los lenguajes existentes en la época se decidió que ninguno los cumplía totalmente, por lo que se hizo un concurso público al que se presentaron cuatro equipos, cuyas propuestas se nombraron con un color: Rojo (Intermetrics), Verde (CII Honeywell Bull), Azul (SofTEch) y Amarillo (SRI International). Finalmente en mayo de 1979 se seleccionó la propuesta Verde diseñada por Jean Ichbiah de CII Honeywell Bull, y se le dio el nombre de Ada. Esta propuesta era un sucesor de un lenguaje anterior de este equipo llamado LIS y desarrollado durante los años 1970.

El nombre se eligió en conmemoración de lady Ada Augusta Byron (1816-1852) Condesa de Lovelace, hija del poeta Lord George Byron, a quien se considera la primera programadora de la Historia, por su colaboración y relación con Charles Babbage, creador de la máquina analítica.

El lenguaje se convirtió en un estándar de ANSI en 1983 (ANSI/MIL-STD 1815) y un estándar ISO en 1987 (ISO-8652:1987).

El DoD y los ministerios equivalentes de varios países de la OTAN exigían el uso de este lenguaje en los proyectos que contrataban (el Ada mandate). La obligatoriedad en el caso de Estados Unidos terminó en 1997, cuando el DoD comenzó a usar productos COTS (commercial off the shelf).

Ada ha sido utilizado por la NASA en sistemas VAX.

## *C) CARACTERISTICAS*

- La sintaxis, inspirada en Pascal, es bastante legible incluso para personas que no conozcan el lenguaje. Es un lenguaje que no escatima en la longitud de las palabras clave, en la filosofía de que un programa se escribe una vez, se modifica decenas de veces y se lee miles de veces (legibilidad es más importante que rapidez de escritura).
- Identificadores y palabras claves son equivalentes sea cual sea el uso de mayúsculas y minúsculas, es decir es un lenguaje case-insensitive.
- En este caso, todo el programa es un único procedimiento, que puede contener subprogramas (procedimientos o funciones) (en este caso: la función Ack).
- Cada sentencia se cierra con end qué\_cerramos. Es un modo de evitar errores y facilitar la lectura. No es necesario hacerlo en el caso de subprogramas, aunque

todos los manuales lo aconsejan y casi todos los programadores de Ada lo hacen.

- El operador de asignación es :=, el de igualdad =. A los programadores de C y similares les puede confundir este rasgo inspirado en Pascal.
- La sintaxis de atributos predefinidos es Objeto'Atributo (o Tipo'Atributo) (nota: esto sólo aplica a atributos predefinidos por el lenguaje, ya que no es el concepto de atributo típico de OOP).
- Se distingue entre "procedimientos" (subrutinas que no devuelven ningún valor pero pueden modificar sus parámetros) y "funciones" (subrutinas que devuelven un valor y no modifican los parámetros). Muchos lenguajes de programación no hacen esta distinción. Las funciones de Ada favorecen la seguridad al reducir los posibles efectos colaterales, pues no pueden tener parámetros in out

### 3) ALGOL:

#### A) EJEMPLO:

```
procedure Absmax(a) Dimensiones:(n, m) Resultado:(y) Subíndices:(i, k);  
value n, m; array a; integer n, m, i, k; real y;  
comment De la matriz a se toma el elemento con el valor absoluto mayor y se coloca en y.  
Los subíndices del elemento se colocan en i y k;  
begin integer p, q;  
y := 0; i := k := 1;  
for p := 1 step 1 until n do  
  for q := 1 step 1 until m do  
    if abs(a[p, q]) > y then  
      begin y := abs(a[p, q]);  
        i := p; k := q  
      end  
    end Absmax
```

#### B) ALGOL W:

Lenguaje elaborado diseñado por Niklaus Wirth y Tony Hoare a partir de los trabajos del grupo ALGOL de la IFIP. Se trata de un lenguaje conciso, simple de implementar, que evita todos los defectos conocidos del lenguaje Algol e incluye sus propias características adicionales. Sin embargo, el grupo Algol no lo adoptó como sucesor de Algol prefiriendo en su lugar al que terminó siendo Algol 68. Algol W fue utilizado por gran cantidad de usuarios y sembró el camino para el nacimiento del lenguaje Pascal.

Entre las características del lenguaje se destacan: Aritmética de doble precisión, números complejos, Strings y estructuras de datos dinámicas, evaluación por valor, pasaje de parámetros por valor, valor resultado o resultado.

### *C)ALGOL 68:*

La definición del lenguaje fue presentada en la reunión del comité ALGOL de la IFIP en 1965. Luego de varios años de revisión del diseño se llegó a una versión definitiva en 1968. Al principal autor es Adriaan Van Wijngarden.

Los objetivos principales de ALGOL 68 son el permitir comunicar algoritmos, el permitir una eficiente ejecución de los mismos en diferentes arquitecturas y el de servir como herramienta para la enseñanza.

Una característica interesante de ALGOL 68 es que su semántica fue definida formalmente antes de ser implementado en base al formalismo llamado gramáticas de dos niveles.

## **4) BASIC:**

### *A) EJEMPLO:*

#### **Ejemplo 1: Saludo**

```
PRINT "Hola"
```

#### **Ejemplo 2: BASIC original no estructurado (Applesoft BASIC)**

```
10 INPUT "¿Cuál es su nombre? "; U$
20 PRINT "Hola "; U$
30 INPUT "¿Cuántos asteriscos desea? "; N
35 S$ = ""
40 FOR I = 1 TO N
50 S$ = S$ + "*"
55 NEXT I
60 PRINT S$
70 INPUT "¿Desea más asteriscos? "; A$
80 IF LEN(A$) = 0 GOTO 70
90 A$ = LEFT(A$, 1)
100 IF (A$ = "S") OR (A$ = "s") THEN GOTO 30
110 PRINT "Adiós ";
120 FOR I = 1 TO 200
130 PRINT U$; " ";
140 NEXT I
150 PRINT
```

### Ejemplo 3: BASIC estructurado moderno

```
INPUT "¿Cuál es su nombre?"; NombreUsuario$
PRINT "Hola "; NombreUsuario$
DO
  INPUT "¿Cuántos asteriscos desea?"; NoAsteriscos
  Asteriscos$ = ""
  Asteriscos$ = REPEAT$("*", NoAsteriscos)
  PRINT Asteriscos$
DO
  INPUT "¿Desea más asteriscos?"; Respuesta$
  LOOP UNTIL Respuesta$ <> ""
LOOP WHILE UCASE$(LEFT$(Respuesta$, 1)) = "S"
PRINT "Adiós";
FOR I = 1 TO 200
  PRINT NombreUsuario$; " ";
NEXT I
PRINT
```

### B) NACIMIENTO Y PRIMEROS AÑOS:

El lenguaje BASIC original fue inventado en 1964 por John George Kemeny (1926-1993) y Thomas Eugene Kurtz (1928-) en el Dartmouth College. En los años subsiguientes, mientras que otros dialectos de BASIC aparecían, el BASIC original de Kemeny y Kurtz era conocido como *BASIC Dartmouth*.

BASIC fue diseñado para permitir a los estudiantes escribir programas usando terminales de computador de tiempo compartido. BASIC estaba intencionado para facilitar los problemas de complejidad de los lenguajes anteriores, con un nuevo lenguaje diseñado específicamente para la clase de usuarios que los sistemas de tiempo compartido permitían: un usuario más sencillo, a quien no le interesaba tanto la velocidad, sino el hecho de ser capaz de usar la máquina. Los diseñadores del lenguaje también querían que permaneciera en el dominio público, lo que contribuyó a que se diseminara.

Los ocho principios de diseño de BASIC fueron:

1. Ser fácil de usar para los principiantes.
2. Ser un lenguaje de propósito general.
3. Permitir que los expertos añadieran características avanzadas, mientras que el lenguaje permanecía simple para los principiantes.
4. Ser interactivo.
5. Proveer mensajes de error claros y amigables.
6. Responder rápido a los programas pequeños.
7. No requerir un conocimiento del hardware de la computadora.
8. Proteger al usuario del sistema operativo.

El lenguaje fue en parte basado en FORTRAN II y en parte en Algol 60, con adiciones para hacerlo apropiado para tiempo compartido y aritmética de matrices, BASIC fue implementado por primera vez en la mainframe GE-265<sup>[2]</sup>, que soportaba múltiples terminales. Contrario a la creencia popular, era un lenguaje compilado al momento de

su introducción. Casi inmediatamente después de su lanzamiento, los profesionales de computación comenzaron a alegar que BASIC era muy lento y simple. Tal argumento es un tema recurrente en la industria de las computadoras.

Aún así, BASIC se expandió hacia muchas máquinas, y se popularizó moderadamente en los minicomputadores como la serie DEC PDP y la Data General Nova. En estos casos, el lenguaje era implementado como un intérprete, en vez de un compilador, o alternativamente, de ambas formas. En estos últimos años se ha dejado de utilizar pasando a ser Visual Basic.

## C) VISUAL BASIC:

### 1- Características generales:

Es un lenguaje de fácil aprendizaje (pero algunos son más complicados debido a que la persona que lo usa tenga conocimiento de este programa amplio) pensado tanto para programadores principiantes como expertos, guiado por eventos, y centrado en un motor de formularios poderoso que facilita el rápido desarrollo de aplicaciones gráficas. Su principal innovación, que luego fue adoptada por otros lenguajes, fue el uso de un tipo de dll, llamado inicialmente vbx y posteriormente ocx, que permiten contener toda la funcionalidad de un control y facilitar su rápida incorporación a los formularios.

Su sintaxis, derivada del antiguo BASIC, ha sido ampliada con el tiempo al agregarse las características típicas de los lenguajes estructurados modernos. Se ha agregado una implementación limitada de la Programación Orientada a Objetos (los propios formularios y controles son objetos), aunque si que admite el polimorfismo mediante el uso de los Interfaces no admite la herencia. No requiere de manejo de punteros y posee un manejo muy sencillo de cadenas de caracteres. Posee varias bibliotecas para manejo de bases de datos, pudiendo conectar con cualquier base de datos a través de ODBC (Informix, DBase, Access, MySQL, SQL Server, PostgreSQL, etc) a través de ADO.

Es utilizado principalmente para aplicaciones de gestión de empresas, debido a la rapidez con la que puede hacerse un programa que utilice una base de datos sencilla, además de la abundancia de programadores en este lenguaje.

### 2- Ventajas

- Permite programar un microcontrolador de forma BASIC\*
- Visual Basic es un lenguaje **simple** y por tanto **fácil de aprender**.
- Su **mayor facilidad** radica en el **dibujado de formularios**, mediante el **arrastre de controles**.
- La **sintaxis** es cercana al **lenguaje humano**.
- Es un lenguaje RAD, centrado en conseguir en el menor tiempo posible los resultados deseados, por eso mismo su mayor uso está en las **pequeñas aplicaciones**, como gestión de bares, empresas, restaurantes...
- Tiene una **ligera implementación de POO**
- Permite el **tratamiento de mensajes de Windows**.

- Gran parte del trabajo en el diseño de formularios está realizado, gracias a la **gran gama de controles incorporados** junto al lenguaje que ahorran costes de tiempo de desarrollo.
- Soporta el uso de componentes **COM** y **ActiveX**.
- Permite crear **controles personalizados** fácilmente del **mismo modo** que el diseño de formularios.
- Permite **generar librerías dinámicas** (DLL) **ActiveX** de forma nativa y **Win32** (no ActiveX, sin interfaz COM) mediante una reconfiguración de su enlazador en el proceso de compilación.

## 5) JAVA

### A) *HISTORIA RECIENTE*

#### **-En el cliente**

La capacidad de los navegadores Web para ejecutar applets de Java ha asegurado la continuidad del uso de Java por el gran público. Flash está más extendido para animaciones interactivas y los desarrolladores están empezando a usar la tecnología AJAX también en este campo. Java suele usarse para aplicaciones más complejas como la zona de juegos de Yahoo, Yahoo! Games, o reproductores de video

#### **-En el servidor**

En la parte del servidor, Java es más popular que nunca, con muchos sitios empleando páginas JavaServer, conectores como Tomcat para Apache y otras tecnologías Java.

#### **-En el PC de escritorio**

Aunque cada vez la tecnología Java se acerca más y más al PC de sobremesa, las aplicaciones Java han sido relativamente raras para uso doméstico, por varias razones.

- Las aplicaciones Java pueden necesitar gran cantidad de memoria física.
- La Interfaz Gráfica de Usuario (GUI) no sigue de forma estricta la *Guía para la Interfaz Humana'* (Human Interface Guidelines), así como tampoco aquella a la que estamos habitualmente acostumbrados. La apariencia de las fuentes no tiene las opciones de optimización activadas por defecto, lo que hace aparecer al texto como si fuera de baja calidad.
- Las herramientas con que cuenta el JDK no son suficientemente potentes para construir de forma simple aplicaciones potentes. Aunque el uso de herramientas como Eclipse, un IDE con licencia libre de alta calidad, facilita enormemente las tareas de desarrollo.
- Hay varias versiones del Entorno en Tiempo de Ejecución de Java, el JRE. Es necesario tener instalada la versión adecuada. El paquete JRE puede ser de tamaño considerable, 7Mbytes, lo que puede ser un inconveniente a la hora de descargarlo e instalarlo.

- Las aplicaciones basadas en la Web están tomando la delantera frente a aquellas que funcionan como entidades independientes. Las nuevas técnicas de programación producen aplicaciones basadas en un modelo en red cada vez más potentes.

Sin embargo hay aplicaciones Java cuyo uso está ampliamente extendido, como los NetBeans, el entorno de desarrollo (IDE) Eclipse, y otros programas como LimeWire y Azureus para intercambio de archivos. Java también es el motor que usa MATLAB para el renderizado de la interfaz gráfica y para parte del motor de cálculo. Las aplicaciones de escritorio basadas en la tecnología Swing y SWT (Standard Widget Toolkit) suponen una alternativa a la plataforma .Net de Microsoft.

## *B) LENGUAJE*

- En un sentido estricto, Java no es un lenguaje absolutamente orientado a objetos, a diferencia de, por ejemplo, Ruby o Smalltalk. Por motivos de eficiencia, Java ha relajado en cierta medida el paradigma de orientación a objetos, y así por ejemplo, no todos los valores son objetos.
- Por el contrario, los programadores de C++ pueden caer en la confusión con Java, porque en éste los tipos primitivos son siempre variables automáticas, y los objetos siempre residen en el montículo (heap), mientras que en C++ ambos casos están en manos del programador, usando el operador new.
- El código Java puede ser a veces redundante en comparación con otros lenguajes. Esto es en parte debido a las frecuentes declaraciones de tipos y conversiones de tipo manual (casting). También se debe a que no se dispone de operadores sobrecargados, y a una sintaxis relativamente simple. Sin embargo, J2SE 5.0 introduce elementos para tratar de reducir la redundancia, como una nueva construcción para los bucles “foreach”.
- A diferencia de C++, Java no dispone de operadores de sobrecarga definidos por el usuario. Sin embargo esta fue una decisión de diseño que puede verse como una ventaja, ya que esta característica puede hacer los programas difíciles de leer y mantener.
- Java es un lenguaje basado en un solo paradigma.
- Java no permite herencia múltiple como otros lenguajes. Sin embargo el mecanismo de los interfaces de Java permite herencia múltiple de tipos y métodos abstractos.
- El soporte de Java para patrones de texto y la manipulación de éste no es tan potente como en lenguajes como Perl, Ruby o PHP, aunque J2SE 1.4 introdujo las expresiones regulares.

## C) EJEMPLOS

### 2- Ejemplo 1

```
// Hola.java
import java.applet. Applet;
import java.awt. Graphics;

public class Hola extends Applet {
    public void paint(Graphics gc) {
        gc.drawString("Hola, mundo!", 65, 95);
    }
}
```

### 1- Ejemplo 2

```
<!-- Hola.html -->
<html>
  <head>
    <title>Applet Hola Mundo</title>
  </head>
  <body>
    <applet code="Hola" width="200" height="200">
    </applet>
  </body>
</html>
```

## 6) PYTHON

### A) INTRODUCCION

**Python** es un lenguaje de programación habitualmente comparado con TCL, Perl, Scheme, Java y Ruby. Actualmente, Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation. La última versión estable del lenguaje es actualmente (Septiembre de 2006) la 2.5. Guido van Rossum, más conocido como Guido, creó Python, un lenguaje de programación de scripting, la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a éste mucho más limpio y elegante para programar.

Python permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK, Qt entre otros...

Python es un lenguaje interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa. También es una calculadora muy útil.

El nombre del lenguaje proviene de la afición de su creador original, Guido van Rossum, por los geniales humoristas británicos Monty Python. El principal objetivo que persigue este lenguaje es la facilidad, tanto de lectura, como de diseño.

## *B) EJEMPLOS:*

El siguiente ejemplo saludará al usuario si se conoce su plato preferido, o le preguntará cuál es en caso contrario.

### **1-ejemplo 1**

```
# -*- coding: iso8859-15 -*-
import sys

# diccionario de comidas preferidas de cada persona
comidas = {
    "Juan" : "tiburón en escabeche",
    "Pablo" : "Paella don Beto",
    "Alfredo" : "sesos de mono",
    "Fulano" : "cucarachas fritas",
    "Sonia" : "burros",
    "Darinel" : "Hamburguesas Mr Paul",
    "Jose" : "Ceviche"
}
afirmativas = set('si','s','mucho',"porfa")
```

### **2-ejemplo 2**

```
def saluda(nombre, comida):
    print "Hola " + nombre + ", ¿te gusta " + comida + "?"
    respuesta = sys.stdin.readline()
    if respuesta not in afirmativas:
        pregunta(nombre)
```

### **3- ejemplo 3**

```
def pregunta(nombre):
    print "Hola " + nombre + ", ¿qué comida te gusta?"
    respuesta = sys.stdin.readline()
    comidas[nombre] = respuesta
```

### **4- ejemplo4**

```
for nombre in ["Juan", "Pablo", "Alfredo", "Fulano", "Sonia",
"Darinel", "Mengano"]:
    if nombre in comidas and comidas[nombre] is not None:
        saluda(nombre, comidas[nombre])
    else:
        pregunta(nombre)
```

## 5- ejemplo 5

```
#importamos el módulo string util para el manejo de cadenas, usaremos
la función split
import string

archivo = raw_input("Digite la ruta del archivo a leer: ")
abrir_archivo = open(archivo, 'r')
leer_archivo = abrir_archivo.read()
texto = string.split(leer_archivo)
contar = 0
```

## 6- ejemplo 6

```
for palabra in texto:
    invertir = ""
    for letra in palabra:
        invertir = letra + invertir
    if invertir == palabra and len(invertir) > 1:
        contar = contar + 1
    print palabra
```

## 7- ejemplo 7

```
if contar == 0:
    print "No hay palabras palíndromas en el archivo"
else:
    print "Palabras palíndromas encontradas: ", contar
abrir_archivo.close()
```

# 7) PASCAL

## A) *CARACTERISTICAS*

A diferencia de lenguajes de programación descendientes de C, Pascal utiliza el símbolo := para la asignación en vez de =. Si bien el segundo es más conciso, la práctica ha demostrado que muchos usuarios utilizan el símbolo de igualdad para comparar valores en lugar del comparador de C que es el símbolo ==. Esta sintaxis conduce a muchos errores o *bugs* difíciles de rastrear en código C. Dado que Pascal no permite dentro de expresiones y utiliza sintaxis distinta para asignaciones y comparaciones, no sufre estos errores.

Otra diferencia importante es que en Pascal, el tipo de una variable se fija en su definición; la asignación a variables de valores de tipo incompatible no están autorizadas (En C, en cambio, el compilador hace el mejor esfuerzo para dar una interpretación a casi todo tipo de asignaciones). Esto previene errores comunes donde variables son usadas incorrectamente porque el tipo es desconocido. Esto también evita

la necesidad de notación húngara, esto es prefijos que se añaden a los nombres de las variables y que indican su tipo.

### *B) EJEMPLO:*

```
program raiz(input, output);
(*
  Obtener la raiz cuadrada de un numero real x.
  Se supone que x no es negativo.
*)
var x, y: real;

begin
  writeln('** Calcular la raiz cuadrada de x **');
  writeln;
  write('Entrar x (> 0): '); readln(x);
  y := sqrt(x);
  writeln;
  writeln('La raiz cuadrada de ', x, ' es ', y);
  writeln; writeln('** Fin **')
  readln; (* Espera a que el usuario pulse enter para salir del
programa *)
end.
```

### *C) COMPILADORES DISPONIBLES:*

Varios compiladores de Pascal están disponibles para el uso del público en general:

- Compilador GNU Pascal (GPC), escrito en C, basado en GNU Compiler Collection (GCC). Se distribuye bajo licencia GPL.
- Free Pascal está escrito en Pascal (el compilador está creado usando FreePascal), es un compilador estable y potente. También distribuido libremente bajo la licencia GPL. Este sistema puede mezclar código Turbo Pascal con código Delphi, y soporta muchas plataformas y sistemas operativos.
- Turbo Pascal fue el compilador Pascal dominante para PCs durante los años 1980 y hasta principios de los años 1990, muy popular debido a sus magníficas extensiones y tiempos de compilación sumamente cortos. Actualmente, versiones viejas de Turbo Pascal (hasta la 5.5) están disponibles para descargarlo gratuito desde el sitio de Borland (es necesario registrarse)
- Delphi es un producto tipo RAD (Rapid Application Development) de Borland. Utiliza el lenguaje de programación Delphi, descendiente de Pascal, para crear aplicaciones para la plataforma Windows. Las últimas versiones soportan compilación en la plataforma .NET.
- Kylix es la versión más nueva de Borland reiterando la rama de Pascal de sus productos. Es descendiente de Delphi, con soporte para el sistema operativo Linux y una librería de objetos mejorada (CLX). El compilador y el IDE están disponibles para uso no comercial. Actualmente este proyecto está discontinuado

## 8) C++

### A) *INTRODUCCION*

C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

Es un lenguaje híbrido, que se puede compilar y resulta más sencillo de aprender para los programadores que ya conocen C. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes como ROOT (enlace externo). Las principales características del C++ son el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica (*templates*). Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución (*RTTI*)

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.

### B) *PRINCIPIOS*

Todo programa en C++ debe tener la función `main()` (a no ser que se especifique en tiempo de compilación otro punto de entrada, que en realidad es la función que tiene el `main()` :

```
int main()  
{}
```

La función `main` debe tener uno de los siguientes prototipos:

```
int main()  
int main(int argc, char** argv)
```

La primera es la forma por defecto de un programa que no recibe parámetros ni argumentos. La segunda forma tiene dos parámetros: *argc*, un número describiendo el número de argumentos del programa (incluyendo el nombre del programa mismo), y *argv*, un puntero a un array de punteros, de *argc* elementos, donde el elemento `argv[i]` representa el *i*-ésimo argumento entregado al programa.

El tipo de retorno de `main` es **int**. Al finalizar la función `main`, debe incluirse el valor de retorno (por ejemplo, `return 0;`, aunque el estándar prevé solamente dos posibles valores de retorno: `EXIT_SUCCESS` y `EXIT_ERROR`, definidas en el archivo `cstdlib`), o salir por medio de la función `exit`. Alternativamente puede dejarse en blanco, en cuyo caso el compilador es responsable de agregar la salida adecuada.

## C) CONCEPTO DE CLASE

Los objetos en C++ son abstraídos mediante una Clase. Según el paradigma de la programación orientada a objetos un objeto consta de:

1. Métodos o funciones
2. Variables Miembro

### Ejemplo 1

```
class Punto
{
// Opcional pero recomendado, ya que por defecto los miembros son
'private'.
private:
    // Variable miembro privada
    int id;
protected:
    // Variables miembro protegidas
    int x;
    int y;
public:
    // Constructor
    Punto();
    // Destructor
    ~Punto();
    // Funciones miembro o métodos
    int ObtenerX();
    int ObtenerY();
};
```

### 1- Constructores

Son unos métodos especiales que se ejecutan automáticamente al crear un objeto de la clase. En su declaración no se especifica el tipo de dato que devuelven, y poseen el mismo nombre que la clase a la que pertenecen. Al igual que otros métodos, puede haber varios constructores sobrecargados, aunque no pueden existir constructores virtuales.

Como característica especial a la hora de implementar un constructor, justo después de la declaración de los parámetros, se encuentra lo que se llama "lista de inicializadores". Su objetivo llamar a los constructores de los atributos que conforman el objeto a construir.

Tomando el ejemplo de la Clase Punto, si deseamos que cada vez que se cree un objeto de esta clase las coordenadas del punto sean igual a cero podemos agregar un constructor como se muestra a continuación:

```
// Clase Punto
//-----
---
class Punto
{
public:
    // Constructor
    Punto()// Inicializar la variables miembro
    : x ( 0 )
    , y ( 0 )
    {
    }

    // Coordenadas del punto
    float x;
    float y;
};
//-----
---

// Main para demostrar el funcionamiento de la clase
#include <iostream>
using namespace std;

int main()
{
    Punto MiPunto;
    cout<<"Coordenada X:"<<MiPunto.x<<endl;
    cout<<"Coordenada Y:"<<MiPunto.y<<endl;
    return 0;
}
```

Si compilamos y ejecutamos el anterior programa, obtenemos una salida que debe ser similar a la siguiente:

Coordenada X:0 Coordenada Y:0

## 2-Destructores.

Los destructores son funciones miembro especiales llamadas automáticamente en la ejecución del programa, y que por tanto **no deben ser llamadas explícitamente por el programador**. Su cometido es liberar los recursos computacionales que el objeto de dicha clase haya adquirido en tiempo de ejecución al expirar este.

Los destructores son invocados automáticamente al alcanzar el flujo del programa el fin del ámbito en el que está declarado el objeto.

-Existen dos tipos de destructores pueden ser públicos o privados, según si se declaran:

- si es publico se llama desde cualquier parte del programa para destruir el objeto.

- si es privado no se permite la destrucción del objeto por el usuario.

## **9) C (LENGUAJE DE NIVEL MEDIO)**

### *A) INTRODUCCION*

C es un lenguaje de programación creado en 1969 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portable entre plataformas y/o arquitecturas. En la práctica, los programadores suelen usar elementos no-portables dependientes del compilador o del sistema operativo.

### *B) VENTAJAS*

- Lenguaje muy eficiente puesto que es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas.
- A pesar de su bajo nivel es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas conocidos.
- Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.

### *C) VARIANTES*

Desde el inicio del lenguaje han surgido varias ramas de evolución que han generado varios lenguajes:

- **Objective-C** es un primer intento de proporcionar soporte para la programación orientada a objetos en C, de escasa difusión, pero actualmente usado en Mac OS X y GNUstep.

- C++ diseñado por Bjarne Stroustrup fue el segundo intento de proporcionar orientación a objetos a C y es la variante más difundida y aceptada. Esta versión combina la flexibilidad y el acceso de bajo nivel de C con las características de la programación orientada a objetos como abstracción, encapsulación y ocultación.

También se han creado numerosos lenguajes inspirados en la sintaxis de C, pero que no son compatibles con él:

- Java, que une la sintaxis del C a una orientación a objetos más similar a la de Smalltalk y Objective C.
- JavaScript, un lenguaje de scripting creado en Netscape e inspirado en la sintaxis de Java diseñado para dar a las páginas web mayor interactividad. A la versión estandarizada se la conoce como ECMAScript.
- C# (pronunciado *C Sharp*) es un lenguaje desarrollado por Microsoft derivado de C/C++ y Java.

## D) EJEMPLO

El siguiente programa imprime en pantalla la frase "Hola Mundo".

```
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hola, Mundo\n");
    return 0;
}
```

## 10) C#

### A) CONCEPTO

C# (pronunciado "si sharp" o **C sostenido**) es un lenguaje de programación orientado a objetos desarrollado por Microsoft y estandarizado, como parte de su plataforma .NET.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes. C# fue diseñado para combinar el control a bajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic.

C# significa, "do sostenido" (C corresponde a do en la terminología musical anglosajona). El símbolo # viene de sobreponer "++" sobre "++" y eliminar las separaciones, indicando así su descendencia de C++.

C#, como parte de la plataforma .NET, está normalizado por ECMA desde diciembre de 2001 (ECMA-334 "Especificación del Lenguaje C#"). El 7 de noviembre de 2005 acabó la beta y salió la versión 2.0 del lenguaje que incluye mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales y tipos anulables. Ya existe la

versión 3.0 de C# en fase de beta destacando los tipos implícitos y el LINQ (Language Integrated Query).

Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones; mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Aunque aún no existen, es posible implementar compiladores que no generen programas para dicha plataforma, sino para una plataforma diferente como Win32 o UNIX.

En la actualidad existen los siguientes compiladores para el lenguaje C#:

- Microsoft Visual C# versión 2002, 2003 y 2005.
- Mono, es una implementación GPL de todo el entorno .NET desarrollado por Novell. Como parte de esta implementación se incluye un compilador de C#.
- Delphi 2006, de Borland Software Corporation.
- dotGNU Portable.NET, de la Free Software Foundation.

***FIN***

***GUILLERMO CAMBRONERO PEREZ***

***1ºBTO Nº 5***