Lenguaje de programación C

L.I. Francisco Ruiz Sala
Instituto de Astronomía
Facultad de Ciencias
UNAM

¿Que es un leguaje de programación?

Es el desarrollo de un algoritmo mediante símbolos y reglas sintácticas y semánticas, expresadas en forma de instrucciones y relaciones lógicas, mediante las cuales se construye el código fuente de una aplicación o pieza de software determinado.
Así, puede llamarse también lenguaje de programación al resultado final de estos procesos creativos.

Tipos de Lenguajes de programación

Lenguajes de bajo nivel:

Se trata de lenguajes de programación que están diseñados para un hardware específico y que por lo tanto no pueden migrar o exportarse a otros computadores.

Sacan el mayor provecho posible al sistema para el que fueron diseñados, pero no aplican para ningún otro.

Lenguajes de alto nivel:

Se trata de lenguajes de programación que aspiran a ser un lenguaje más universal, por lo que pueden emplearse indistintamente de la arquitectura del hardware, es decir, en diversos tipos de sistemas.

Los hay de propósito general y de propósito específico.

Clasificacion de Lenguajes de Programación

Lenguajes imperativos:

Menos flexibles, dada la secuencialidad en que construyen sus instrucciones, estos lenguajes programan mediante órdenes condicionales y un bloque de comandos al que retornan una vez llevada a cabo la función o procedimiento. Ejemplos: C, Pascal, Fortran, B, python, Basic, Java, Cobol etc.

Lenguajes funcionales o declarativos:

También llamados procedimentales, estos lenguajes programan mediante funciones que son invocadas conforme a la entrada recibida, que a su vez son resultado de otras funciones. Ejemplos: SQL, Hskel, Prolog

https://concepto.de/lenguaje-de-programacion/

Definición: Lenguaje C

El lenguaje C 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 portátil entre plataformas y/o arquitecturas.

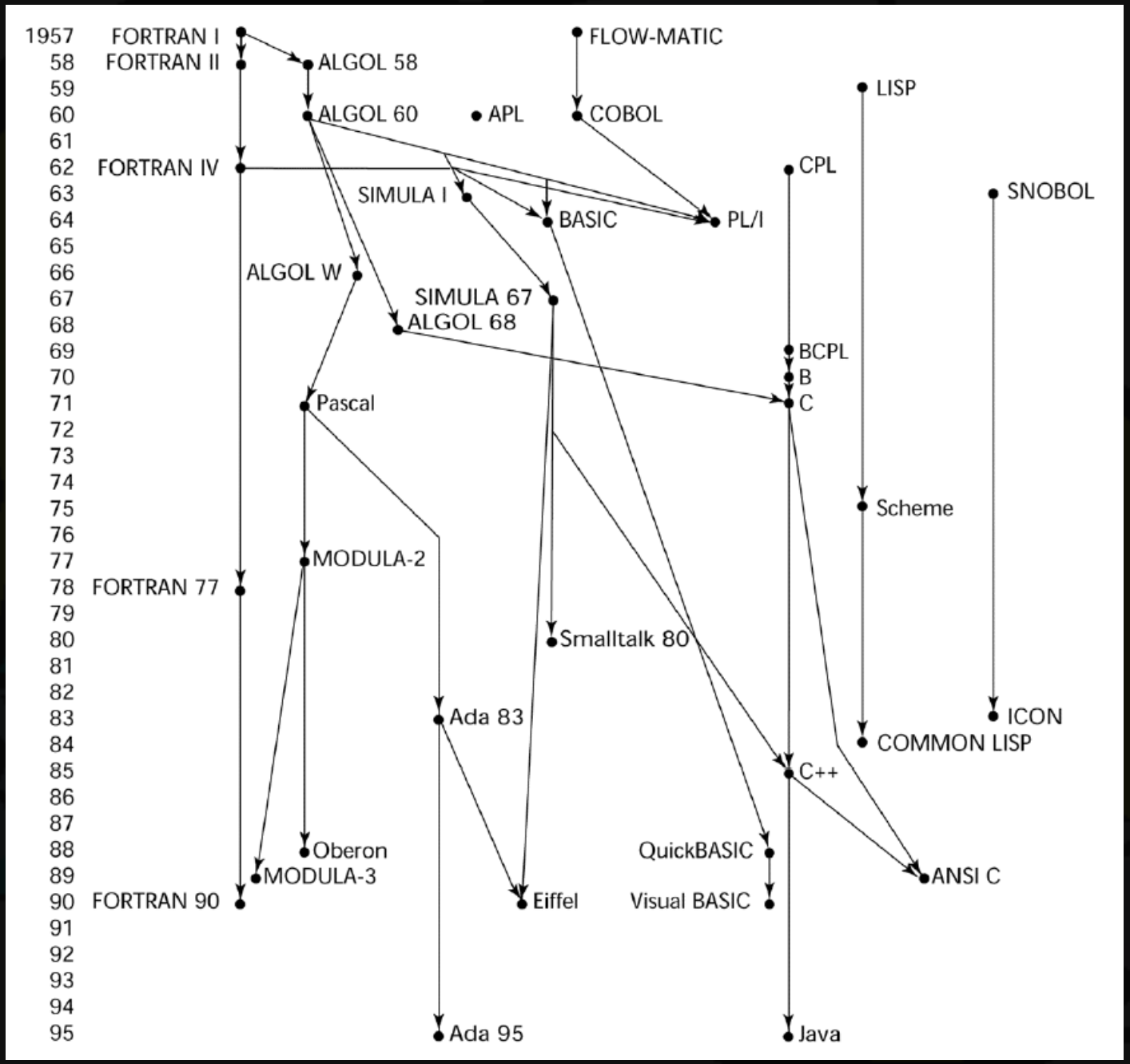
Historia:

C es un lenguaje de programación creado en 1972 por 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.

Historia:

La primera estandarización del lenguaje C fue en ANSI (American National Standards Institute). El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO. La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portátil entre plataformas y/o arquitecturas.


1957
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
FORTRAN I
FORTRAN II
ALGOL 58
FLOW-MATIC
LISP
ALGOL 60
APL
COBOL
FORTRAN IV
CPL
SNOBOL
SIMULA I
BASIC
PL/I
ALGOL W
SIMULA 67
ALGOL 68
BCPL
B
C
Pascal
Scheme
MODULA-2
FORTRAN 77
Smalltalk 80
Ada 83
ICON
COMMON LISP
C++
Oberon
QuickBASIC
MODULA-3
ANSI C
FORTRAN 90
Eiffel
Visual BASIC
Ada 95
Java

Caracteristicas:

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, 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.

Caracteristicas:

- Núcleo de lenguaje simple que incluye funciones matemáticas y de manejo de archivos proporcionadas por bibliotecas
- Lenguaje flexible que permitía programar de manera estructurada y no estructurada
- Inclusión de un sistema de tipos que impedía la realización de operaciones sin sentido
- Acceso a memoria de bajo nivel mediante el uso de punteros
- Conjunto reducido de palabras reservadas
- Paso de parámetros a funciones por valor
- Tipos de datos agregados (struct) que permitían combinar, bajo un mismo tipo, datos de tipos distintos y manejarlos como uno solo.

-Es muy portable (transportable entre un gran número de plataformas hardware y plataformas software, sistemas operativos).Existen numerosos compiladores para todo tipo de plataformas sobre los que corrren los mismos programas fuentes o con ligeras modificaciones.

+Es versátil y de bajo nivel, por lo que es idóneo para tareas relativas a la programación del sistema.

-A pesar de ser un excelente lenguaje para programación de sistemas, es también un eficiente y potente lenguaje para aplicaciones de propósito general.

-Es un lenguaje pequeño, por lo que es relativamente fácil construir compiladores de C y además es también fácil de aprender.

El lenguaje C ofrecía diversos tipos de datos: "CHAR" (carácter), "INT" (enteros), "FLOAT" (números reales en precisión simple), y "DOUBLE" (números reales en precisión doble) a los que posteriormente se añadieron los tipos "SHORT" (números enteros de tamaño menor a un INT), "LONG" (números enteros de tamaño mayor a un INT), "UNSIGNED" (Enteros sin signo), y "ENUMERACIONES" que junto a las estructuras, las uniones y las tablas y matrices componían un variado juego de tipos de datos con los que abordar múltiples problemas.

-Todos los compiladores suelen incluir potentes y excelentes bibliotecas de funciones compatibles con el estándar ANSI. Los diferentes fabricantes suelen añadir a sus compiladores funcionalidades diversas que aumentan la eficiencia y potencia de los mismos y constituye una notable ventaja respecto a otros lenguajes.

-Es un lenguaje muy utilizado para la construcción de: sistemas operativos, ensambladores, programas de comunicaciones, intérpretes de lenguajes, compiladores de lenguajes, editores de textos, bases de datos, utilidades, controladores de red, etc.

¿Que es un lenguaje ensamblador?

El lenguaje ensamblador o assembler (en inglés: assembly language y la abreviación asm) es un lenguaje de programación de bajo nivel. Consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Wikipedia

```
; ---------------------------------------------
; Programa que imprime un string en la pantalla
; ---------------------------------------------
        .model small            ; modelo de memoria
        .stack                  ; segmento del stack
        .data                   ; segmento de datos
        Cadena1 DB 'Hola Mundo.$' ; string a imprimir (finalizado en $)
        .code                   ; segmento del código
; ---------------------------------------------
; Inicio del programa
; ---------------------------------------------
        programa:
                ; ---------------------------------------------------------------------------------------
                ; inicia el segmento de datos
                ; ---------------------------------------------------------------------------------------
                MOV AX, @data           ; carga en AX la dirección del segmento de datos
                MOV DS, AX              ; mueve la dirección al registro de segmento por medio de AX
                ; ---------------------------------------------------------------------------------------
                ; Imprime un string en pantalla
                ; ---------------------------------------------------------------------------------------
                MOV DX, offset Cadena1 ; mueve a DX la dirección del string a imprimir
                MOV AH, 9               ; AH = código para indicar al MS DOS que imprima en la pantalla, el string en DS:DX
                INT 21h                 ; llamada al MS DOS para ejecutar la función (en este caso especificada en AH)
                ; ---------------------------------------------------------------------------------------
                ; Finaliza el programa
                ; ---------------------------------------------------------------------------------------
                INT 20h                 ; llamada al MS DOS para finalizar el programa
        end programa
```

¿Que es un Interprete?

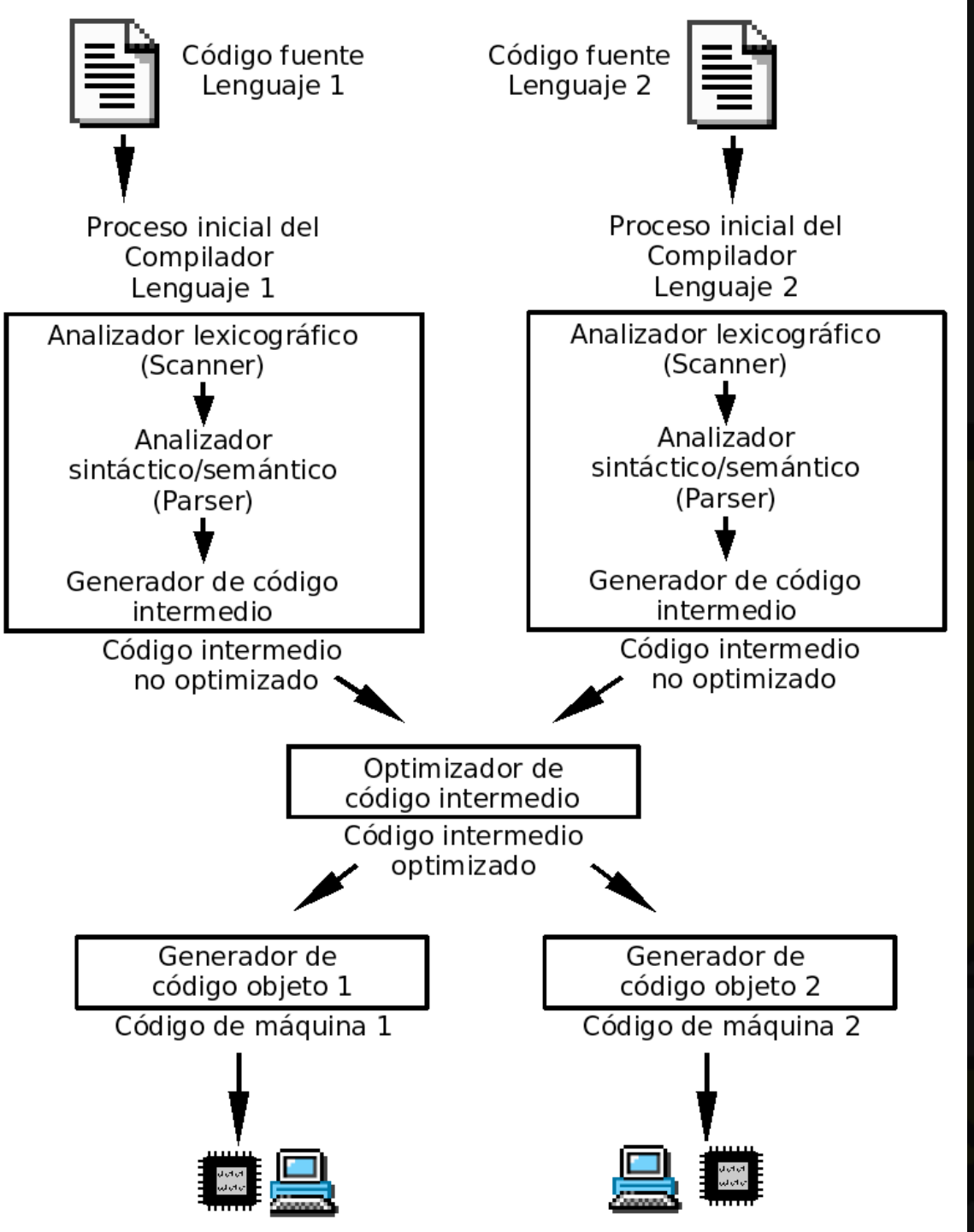
Un intérprete es un programa que ejecuta línea a línea las instrucciones de un lenguaje de alto nivel.
El intérprete carga el código fuente y traduce las instrucciones a un lenguaje intermedio que puede luego ser ejecutado.
Los intérpretes se diferencian de los compiladores o de los ensambladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

Que es un Compilador:

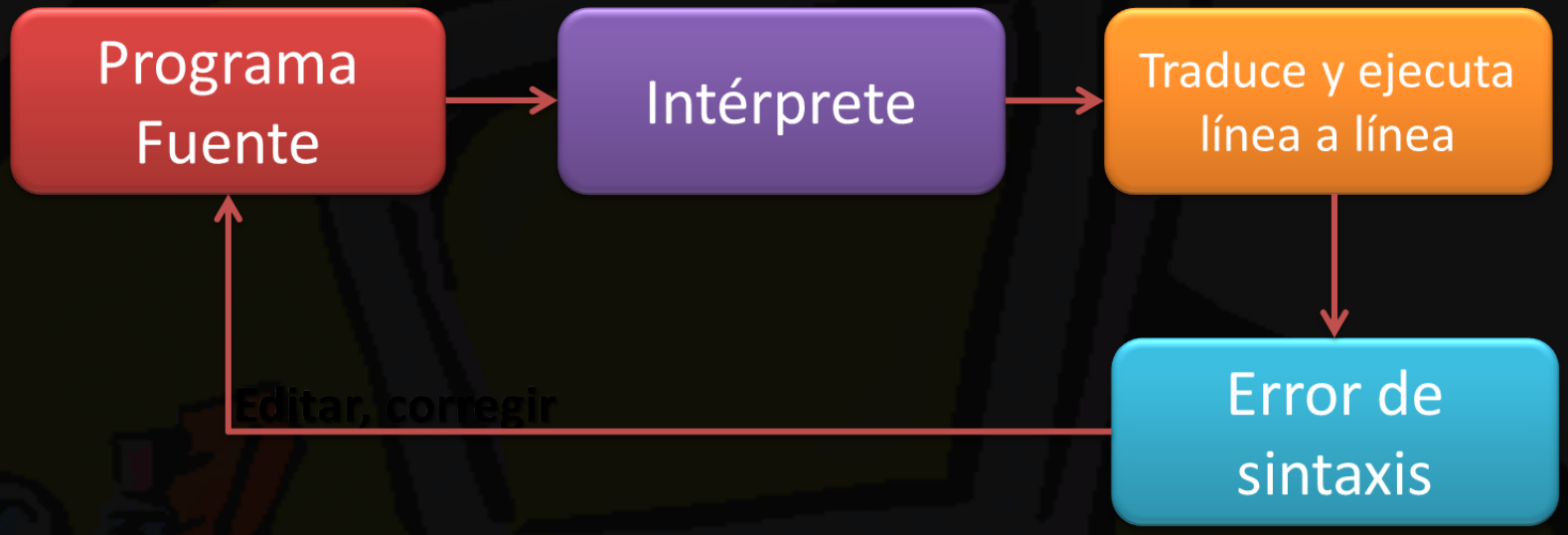
- En informática, un compilador es un programa informático que transforma código fuente escrito en un lenguaje de programación o lenguaje informático (el lenguaje fuente), en otro lenguaje informático (el lenguaje objetivo, estando a menudo en formato binario conocido como código objeto). La razón más común para querer transformar código fuente es crear un programa ejecutable.

- Cualquier programa escrito en un lenguaje de programación de alto nivel debe ser traducido a código objeto antes de que pueda ser ejecutado, para que todos los programadores que usen tal lenguaje usen un compilador o un intérprete. Por esto, los compiladores son muy importantes para los programadores. Cualquier mejora hecha a un compilador lleva a un gran número de programas mejorados

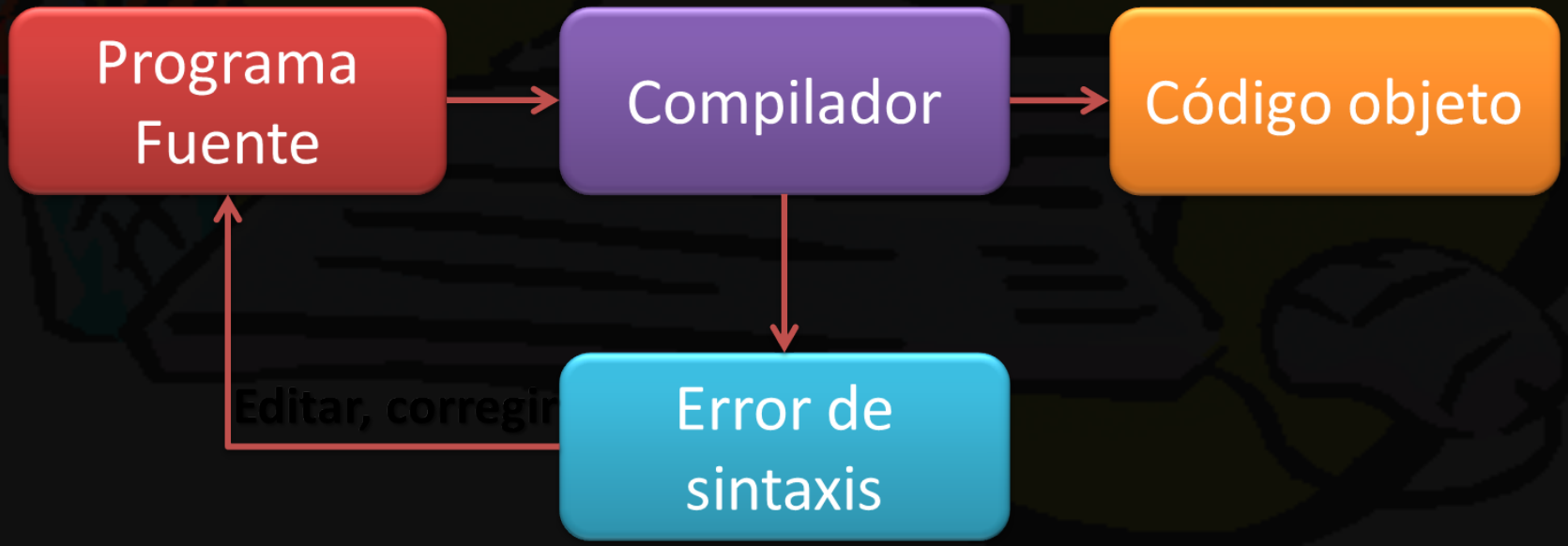
wikipwdia

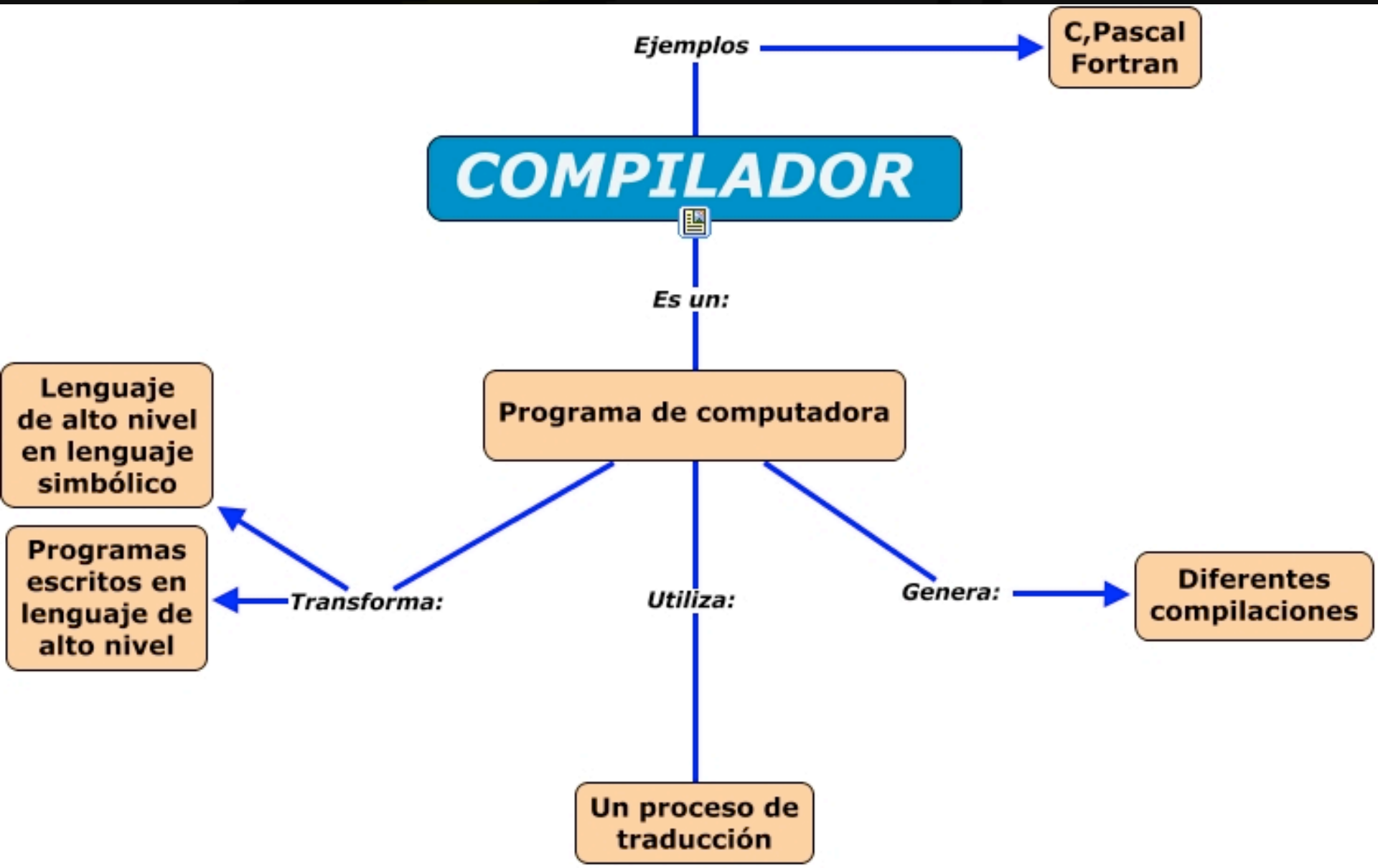

Código fuente
Lenguaje 1
Código fuente
Lenguaje 2
Proceso inicial del
Compilador
Lenguaje 1
Proceso inicial del
Compilador
Lenguaje 2
Analizador lexicográfico
(Scanner)
Analizador
sintáctico/semántico
(Parser)
Generador de código
intermedio
Analizador lexicográfico
(Scanner)
Analizador
sintáctico/semántico
(Parser)
Generador de código
intermedio
Código intermedio
no optimizado
Código intermedio
no optimizado
Optimizador de
código intermedio
Código intermedio
optimizado
Generador de
código objeto 1
Generador de
código objeto 2
Código de máquina 1
Código de máquina 2

Interprete



Compilador




Ejemplos
C,Pascal
Fortran
COMPILADOR
Es un:
Programa de computadora
Lenguaje de alto nivel en lenguaje simbólico
Programas escritos en lenguaje de alto nivel
Transforma:
Utiliza:
Genera:
Diferentes compilaciones
Un proceso de traducción

Modo de traducción

Código fuente

```
{ ...
begin
  writeln('** Calcular la raíz cuadrada de 12 **');
  writeln('Entrar x (> 0): ');
  readln(x);
  y := sqrt(abs(x)); (* Raíz cuadrada del valor absoluto de x para evitar raíces imaginarias *)
  writeln;
  if (x<0) then (* Si x es negativo, el resultado se notifica como imaginario *)
    writeln('La raíz cuadrada de ', x, ' es el número imaginario ', y,'i')
  else
    writeln('La raíz cuadrada de ', x, ' es ', y);
  writeln;
  writeln('** Fin **');
 end.
...}
```

Archivo: miPrimerPrograma.pas

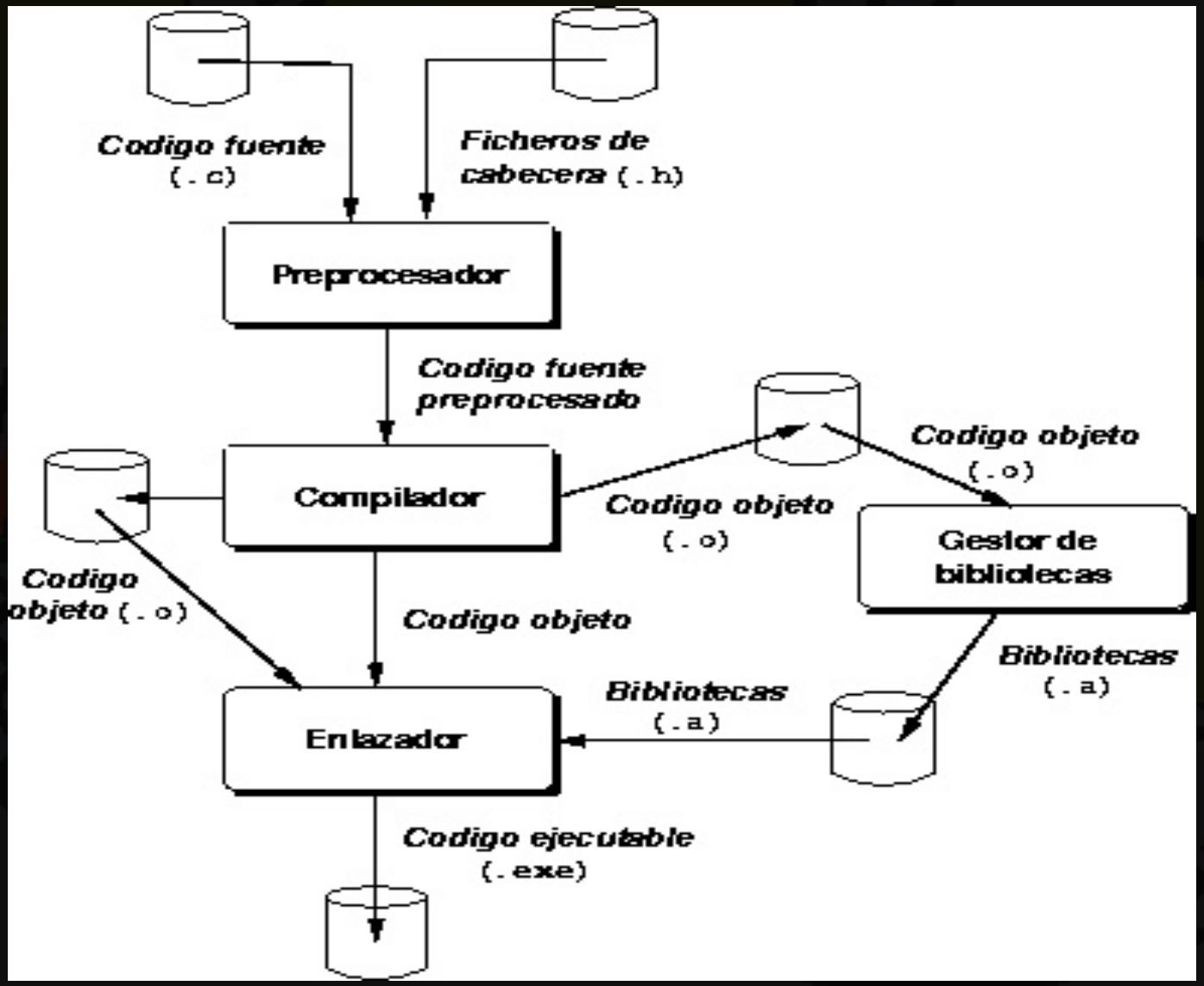
COMPILADOR

Código máquina

```
00000010 10000000 00000000
00000110 2068 10001000 10000000
01000000 00000010 11001010
00000001 00000000 00000000 2080
00010000 10111111 11111111
11111011 10000110 10000000
11000000 00000101 10000001
11000011 11100000 00000100
00000000 00000000 00000000
00010100 00000000 00000000
00001011 10111000 00000010
10000000 00000000 00000110 2068
10001000 10000000 01000000
00000010 11001010 00000001
00000000 00000000 2080 00010000
10111111 11111111 11111011
10000110 10000000 11000000
00000101 10000001 11000011
11100000 00000100 00000000
00000000 00000000 00010100
00000000 00000000 00001011
10111000
```

Archivo: miPrimerPrograma.exe

Formato de compilador



ESTRUCTURA GENERAL DE UN PROGRAMA EN C

Un programa en C se compone de una o más funciones. Una de las funciones debe ser obligatoriamente **main. Una** función en ***C*** es **un** grupo de instrucciones que realizan una o más acciones. Asimismo, un programa contendrá una serie de directivas **#include** que permitirán incluir en el mismo archivo de cabecera que a su vez constarán de funciones **y** datos predefinidos en ellos.

```
#include        Directivas del preprocesador

#define         Macros del procesador
```

Declaraciones globales
- prototipos de funciones
- variables

Función principal `main`

```
main()
{
  declaraciones locales
  sentencias
}
```

Definiciones de otrasfunciones

```
tipo1 func1(...)
{
  ...
}
```

Figura 3.1. Estructura típica de un programa C.

Ejemplos:

```
// <-- Comentario.
// necesario para utilizar printf()
# include <stdio.h>
// Forzosamente se basa en librerias pre-definidas
int main(void)
{
        printf("Hola Mundo\n");
        return 0;
}
```

```c
/*Listado DEMO_UNO.c.Programa de saludo*/

#include<stdio.h>
/*Este programa imprime: Hola mundo*/

int main()
{
  printf("Hola mundo \n");
  return 0;
}
```

Explicacion del programa

-La segunda línea es un comentario, identificado por los caracteres /* y */. Los comentarios se incluyen en programas que proporcionan explicaciones a los lectores de los mismos. Son ignorados por el compilador.
-La tercera línea contiene la cabecera de la función main (, obligatoria en cada programa C. Indica el comienzo del programa y requieren los paréntesis ( ) a continuación de main ( ) .
-La cuarta y séptima línea contienen sólo las llaves { y 1 que encierran el cuerpo de la función main ( ) y son necesarias en todos los programas C.
-La quinta línea contiene la sentencia
printf("Bienvenid0 a la programación en C\n");
que indica al sistema que escriba el mensaje "Bienvenido a la programación en C\n". printfl) eslafunciónmásutilizadaparadarsalidadedatosporeldispositivoestándar,lapantalla.
-La salida será
Bienvenido a la programación en C
+El símbolo ' \n' es el símbolo de nueva línea. Poniendo este símbolo al final de la cadena entre comillas, indica al sistema que comience una nueva línea después de imprimir los caracteres precedentes, terminando, por consiguiente, la línea actual.
-La sexta línea contiene la sentencia return o. Esta sentencia termina la ejecución del programa y devuelve el control al sistema operativo de la computadora. El número O se utiliza para señalar que el programa ha terminado correctamente (con éxito).
-Obsérvese el punto y coma (;) al final de la quinta y sexta línea. C requiere que cada sentencia termine con un punto y coma. No es necesario que esté al final de una línea. Se pueden poner varias sentencias en la misma línea y se puede hacer que una sentencia se extienda sobre varias líneas.

main e include

Un programa en C se compone de una o más funciones. Una de las funciones debe ser obligatoriamente **main. Una** función en ***C*** es **un** grupo de instrucciones que realizan una o más acciones. Asimismo, un programa contendrá una serie de directivas **#include** que permitirán incluir en el mismo archivo de cabecera que a su vez constarán de funciones **y** datos predefinidos en ellos.

```
#include        Directivas del preprocesador

#define         Macros del procesador
```

Declaraciones globales
- prototipos de funciones
- variables

Función principal `main`

```
main()
{
  declaraciones locales
  sentencias
}
```

Definiciones de otrasfunciones

```
tipo1 func1(...)
{
  ...
}
```

Figura 3.1. Estructura típica de un programa C.

Función main()

Cada programa C tiene una función main() que es el punto de entrada al programa. Su estructura es:

```
Main()
{
....        Bloque de sentencias

}
```

Las sentencias incluidas entre las llaves { . . . } se denominan *bloque.* Un programa debe tener sólo una función main (. Si se intenta hacer dos funciones main ( ) se produce un error. Además de la función main ( ) , un programa C consta de una colección de funciones.

Una función *C* es un **subprograma que devuelve** un **único valor, un conjunto de valores.**

En un programa corto, el programa completo puede incluirse totalmente en la función main ( ) . Un programa largo, sin embargo, tiene demasiados códigos para incluirlo en esta función. La función main() en un programa largo consta prácticamente de llamadas a las funciones definidas por el usuario.

Mas Ejemplos

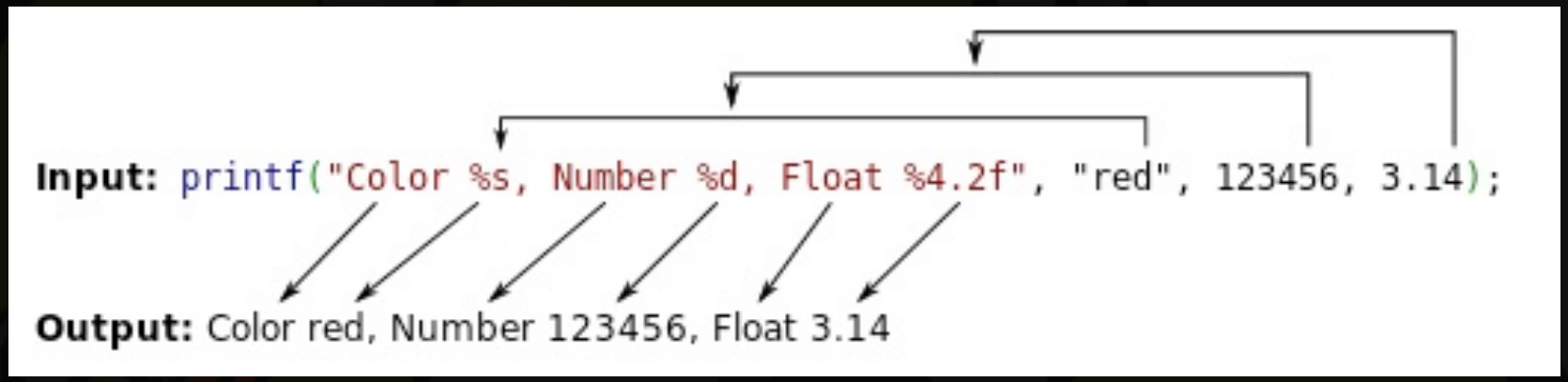
```
/*Listado DEMO_UNO.c.Programa de saludo*/

#include<stdio.h>
/*Este programa imprime: Hola mundo*/

int main()
{
  printf("Hola mundo \n");
  return 0;
}
```

La directiva #include de la primera línea es necesaria para que el programa tenga salida. ***Se*** refiere a un archivo externo denominado stdio.h en el que se proporciona la información relativa a la función printf ( ) . Obsérvese que los ángulos < y > no son parte del nombre del archivo; se utilizan para indicar que el archivo es un archivo de la biblioteca estándar C.

Formato del printf



printf("Color %s, numero1 %d, numero2 %05d, hex %x, real %5.2f.\n", "rojo", 12345, 89, 255, 3.14);

Formato del scanf

scanf (cadena de control, lista de argumentos);

Carácter	Significado
c	Lee un carácter simple
d	Lee un entero decimal
i	Lee un entero decimal, octal o hexadecimal
e	Lee un número en punto flotante
f	Lee un número en punto flotante
g	Usa %e o %f, el más corto en long.
o	Lee un entero octal corto
s	Lee una cadena de caracteres
u	Lee un entero decimal sin signo
x	Lee un entero hexadecimal
[...]	Lee una cadena de caracteres con caracteres de espaciado
p	Lee un puntero

Ejemplos scanf y printf

```
//Ejemplo scanf
#include <stdio.h>
int n;
printf("Introduce un numero: ");
scanf("%d",&n);
```

```
//Ejemplo prinf
#include <stdio.h>
int main(void)
{
    printf("¡Hola, mundo!\n");
    return 0;
}
```

```
/*Entrada y Salida: scanf*/
#include <stdio.h>
int main()
{
    float a;
    char b;
    printf("Introduce un numero: ");
    scanf("%f",&a);
    printf("Intrduciste el numero: %f\n",a);
    getchar();
    printf("Introduce una letra: ");
    scanf("%c",&b);
    printf("Introduciste la letra %c\n\n",b);
    printf("Presiona Enter pasa Salir");
    getchar();
    getchar();
    return 0;

}
```

Palabras reservadas

Una palabra reservada ***(keyword o reserved word), tal*** como void es una característica del lenguaje C asociada con algún significado especial. Una palabra reservada no se puede utilizar como nombre de identificador o función.

Los siguientes identificadores están reservados para utilizarlos como *palabras* ***reservadas,*** y no se deben emplear para otros propósitos.

asm enum auto extern break float case for char goto const if signed sizeof static struct switch typedef continue int union default long un8igned do register void double return volati1e else short while

COMENTARIOS

Ya se ha expuesto antes que los comentarios en ***C*** tienen el formato: /*...*/

Los comentarios se encierran entre / * y * / pueden extenderse a lo largo de varias líneas. / * Titulo: Demo-uno por

Mr. Martinez * /

ARCHIVOS DE CABECERA

Un archivo de cabecera es un archivo especial que contiene declaraciones de elementos y funciones de la biblioteca. Para utilizar macros, constantes, tipos y funciones almacenadas en una biblioteca, un programa debe utilizar la directiva #includepara insertar el archivo de cabecera correspondiente. Por ejemplo, si un programa utiliza la función pow que se almacena en la biblioteca matemática *math.h,* debe contener la directiva

#include <math.h>

para hacer que el contenido de la biblioteca matemática esté disponible a un programa. La mayoría de los programas contienen líneas como ésta al principio, que se incluyen en el momento de compilación.

#include <stdio.h>

TIPOS DE DATOS EN C

Los tres tipos de datos básicos en C son:
Enteros
Números de coma flotante (reales)
carácteres

Tipo	Ejemplo	Tamaño en bytes	Rango Mínimo..Máximo
char	'C'	1	0..255
short	-15	2	-128..127
int	1024	2	-32768..32767
unsigned int	42325	2	0..65535
long	262144	4	-2147483648..2147483637
float	10.5	4	3.4*(10)..3.4*(10)
double	0.00045	8	1.7*(10)..1.7*(10)
long double	1e-8	8	*igual que* double

Declaración de variables

La forma más simple de una declaración de variable en C es poner primero el tipo de dato y a continuación el nombre de la variable. Si se desea dar un valor inicial a la variable, éste se pone a continuación. El formato de la declaración es:

<tipo de dato> <nombre de variable> = <valor inicial> Se pueden también declarar múltiples variables en la misma línea: *<tipo-de-dato> <nom-variz,cnom-var2>* ... *<nom-varn>* Así, por ejemplo,

int longitud; int valor = 99;
int valor1, valor2;
int num-parte = 1141, num-items = 45;

Caracteres secuencia de escape

Un carácter que se lee utilizando una barra oblicua (\) se llama *secuencia* o *código de escape.*

Código de escape	Significado	Códigos ASCII Dec	Hex
'\n'	nueva línea	13 10	0D 0A
'\r'	retorno de carro	13	0D
'\t'	tabulación	9	09
'\v'	tabulación vertical	11	0B
'\a'	alerta (pitido sonoro)	7	07
'\b'	retroceso de espacio	8	08
'\f'	avance de página	12	0C
'\\'	barra inclinada inversa	92	5 c
'\''	comilla simple	39	27
'\"'	doble comilla	34	22
'\?'	signo de interrogación	63	3F
'\000'	número octal	*Todos*	*Todos*
'\xhh'	número hexadecimal	*Todos*	*Todos*

Constantes definidas

Las constantes pueden recibir nombres simbólicos mediante la directiva **#define.**

#define NUEVALINEA \n

#define PI 3.141592

#define VALOR 54

C sustituye los valores \n, 3.141592 y 54 cuando se encuentra las constantes simbólicas NUEVALINEA, P I y VALOR.Las líneas anteriores no son sentencias y, por ello, no terminan en punto y coma.

printf("Elvalor es %dNUEVALINEA",VALOR);
Escribe en pantalla la constante VALOR. Realmente, el compilador lo que hace es sustituir en el

progama todas las ocurrencias de VALOR por 54, antes de analizar sintácticamente el programa fuente.

3.9.3. Constantesenumeradas

Variables

En C una *variable* es una posición con nombre en memoria donde se almacena un valor de un cierto tipo de dato. Las variables pueden almacenar todo tipo de datos: cadenas, números y estructuras. Una *constante,* por el contrario, es una variable cuyo valor no puede ser modificado.

Una variable típicamente tiene un nombre (un identificador) que describe su propósito. Toda variable utilizada en un programa debe ser declarada previamente. La definición en C debe situarse al principio del bloque, antes de toda sentencia ejecutable. Una definición reserva un espacio de almacenamiento en memoria. El procedimiento para definir *(crear)una* variable es escribir el tipo de dato, el identificador o nombre de la variable y, en ocasiones, el valor inicial que tomará. Por ejemplo,

char Respuesta;

significa que se reserva espacio en memoria para Respuesta,en este caso, un carácter ocupa un solo byte.

Declaración de variables

Una *declaración* de una variable es una sentencia que proporciona información de la variable al cornpilador C. Su sintaxis es:

tipo variable

tipo es el nombre de un tipo de dato conocido por el C

variable es un identificador (nombre) válido en **C.**

```
/*Distancia a la luna en kilometros*/
#include<stdio.h>

main()
{
  int luna=238857;                        /*Distancia en millas*/
  float luna_kilo;
  printf("Distancia a la luna %d millas \n",luna);
  luna_kilo=luna*1.609;                    /*una milla=1.609km*/
  printf("En kilometros es %f Km \n",luna_kilo);
}
```

Entradas y salidas

C proporciona facilidades para entrada y salida, para lo que todo programa deberá tener el archivo de cabecera stdio.h . En C la entrada y salida se lee y escribe de los dispositivos estándar de entrada y salida, se denominan stdin y stdout respectivamente. La salida, normalmente, es a pantalla del ordenador, la entrada se capta del teclado. En el archivo stdio.h están definidas macros, constantes, variables y funciones que permiten intercambiar datos con el exterior. A continuación se muestran las más habituales y fáciles de utilizar.

Entrada

```c
#include<stdio.h>
main()
{
  char primero, ultimo;
  printf("Introduzca su primera y última inicial:");
  scanf("%c %c",&primero, &ultimo);
  printf("Hola, %c .%c. \n", primero, ultimo);
}
```

```c
#include<stdio.h>

main()
{
  char nombre[51];
  printf("Nombre: ");
  scanf("%s", nombre);
  printf("Nombre introducido: %s\n", nombre);
}
```

Salida

Los códigos de formato más utilizados y su significado:

`%d`	El dato se convierte a entero decimal.
`%o`	El dato entero se convierte a octal.
`%x`	El dato entero se convierte a hexadecimal.
`%u`	El dato entero se convierte a entero sin signo.
`%c`	El dato se considera de tipo carácter.
`%e`	El dato se considera de tipo `f 1oat`. Se convierte a notación científica, de la forma *{-}n.mmmmmmmE{+|-}dd*.
`%f`	El dato se considera de tipo `float`. Se convierte a notación decimal, con parte entera y los dígitos de precisión.
`%g`	El dato se considera de tipo `float`. Se convierte según el código *%e* o *%f* dependiendo de cual sea la representación más corta.
`%s`	El dato ha de ser una cadena de caracteres.
`%lf`	El dato se considera de tipo `double`.

La sentencia if

La sentencia if funciona de la siguiente manera. Cuando se alcanza la sentencia if dentro de un programa, se evalúa la ***expresión*** entre paréntesis que viene a continuación de **if.** Si ***Expresión*** es verdadera, se ejecuta *Acción;* en caso contrario no se ejecuta *Acción* (en su formato más simple, *A c c i ó n* es una sentencia simple y en los restantes formatos es una sentencia compuesta). En cualquier caso la ejecución del programa continúa con la siguiente sentencia del programa

La sentencia if

if *(condición) sentencia;*

condición es una expresión entera(lógica).

sentencia es cualquier sentencia ejecutable, que se ejecutará sólo si la ***condición*** toma un valor distinto de cero.

```
#include<stdio.h>
#include<math.h>
main()
{
  int n, d;
  printf("Introduzca dos enteros: ");
  scanf("%d %d", &n, &d);
  if(n%d == 0)printf("%d es divisible por %d\n",n,d);
}
```

La sentencia if-else

if *(Expresión)* *Acción* ***else*** *Acción*

En este formato *Acción y Acción son individualmente,* o bien una Única sentencia que termina en un punto y coma (;) o un grupo de sentencias encerrado entre llaves. Cuando se ejecuta la sentencia **if-else, se** evalúa *Expresión.* Si *Expresión* es verdadera, se ejecuta *Acción* y en caso contrario se ejecuta

La sentencia if-else

```
#include<stdio.h>

main()
{
  int n, d;
  printf("Introduzca dos enteros: ");
  scanf("%d %d", &n, &d);
  if (n/d==0)
    printf("%d no es divisible por %d\n",n, d);
  else
    printf("%d no es divisible por %d\n",n,d);
}
```

La sentencia if-else

```
#include<stdio.h>
#include<math.h>
int main()
{
  float f, x;
  printf("\n Elige un valor de x:\t ");
  scanf("%f",&x);
  /*selección del rango en que se encuentra x*/
  if(x<=0.0)
    f=pow(x,2);
  else
 f=-pow(x,2)+3*x;
  printf("f(%.1f)=%.3f\n",x,f);
}
```

La sentencia while

Un **bucle** ***(ciclo)es*** cualquier construcción de programa que repite una sentencia o secuencia de sentencias un número de veces. La sentencia *(o grupo* de sentencias) que se repiten en un bloque se denomina **cuerpo** del bucle y cada repetición del cuerpo del bucle se llama **iteración** del bucle.

La sentencia while

El comportamiento o funcionamiento de una sentencia (bucle) while es:

1. Se evalúa la *condición- bucle*
2.Si *condición-bucle es verdadera* (distinto decero):

a . La *sentencia* especificada, denominada el **cuerpo** del bucle, se ejecuta.

b. Vuelve el control al paso I.

3. En caso contrario:

El control se transfiere a la sentencia siguiente al bucle o sentencia while.

Las sentencias del cuerpo del bucle se repiten mientras que la expresión lógica (condición del bucle) sea verdadera. Cuando se evalúa la expresión lógica y resulta falsa, se termina y se ***sale*** del bucle y se ejecuta la siguiente sentencia de programa después de la sentencia **while.**

La sentencia while

```
/*Visualizar n asteriscos*/
#include<stdio.h>

main()
{
  int i,n;
   printf("Número de asteriscos: \n");
   scanf("%d",&n);

   i=0;   /*inicialización*/
   while(i<n) /*condición*/
  {
    printf("*" );
    ++i;  /*incrementa en 1 contador*/
  } /*fin de while*/
  printf("\n",n);
}
```

La sentencia while

```
#include<stdio.h>
#include<string.h>
main()
{
  int ContadorNum=0;
  int TotalNum=6;
  float SumaNum=0;
  float media;
  printf("Introduzca %d números\n",TotalNum);
  while (ContadorNum<TotalNum)
  {
    /*Valores a procesar*/
    float numero;
    scanf("%f",&numero);
    SumaNum+=numero;
    ++ContadorNum;
  }
  media=SumaNum/ContadorNum;
  printf("Media: %.2f\n",media);
}
```

El bucle for

La sentencia for (bucle for)es un método para ejecutar un bloque de sentencias un número fijo de veces. El bucle for se diferencia del bucle while en que las operaciones de control del bucle se sitúan en un solo sitio: la cabecera de la sentencia.

for (*Inicialización; Condición; Incremento*)

sentencias

El bucle for

```c
#include<math.h>
#include<stdio.h>

#define M 15
#define f(x) exp(2*x)-x

main()
{
  int i;
  double x;
  for(i=1;i<=M;i++)
    {
      printf("Valor de x: ");
      scanf("%lf",&x);
      printf("f(%.1lf)=%.4g\n",x,f(x));
    }
}
```

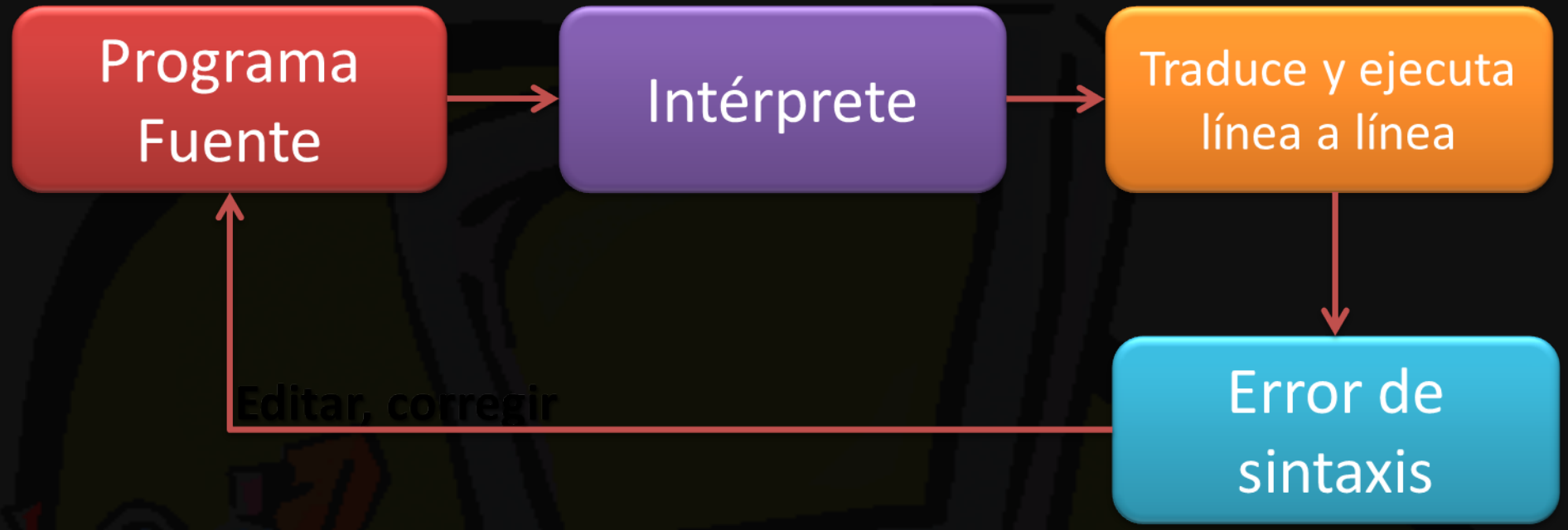
```c
#include<stdio.h>
main()
{
  int n;
  for(n=0;n<100;n+=20)
    printf("%d \t %d\n",n,n*n);
}
```

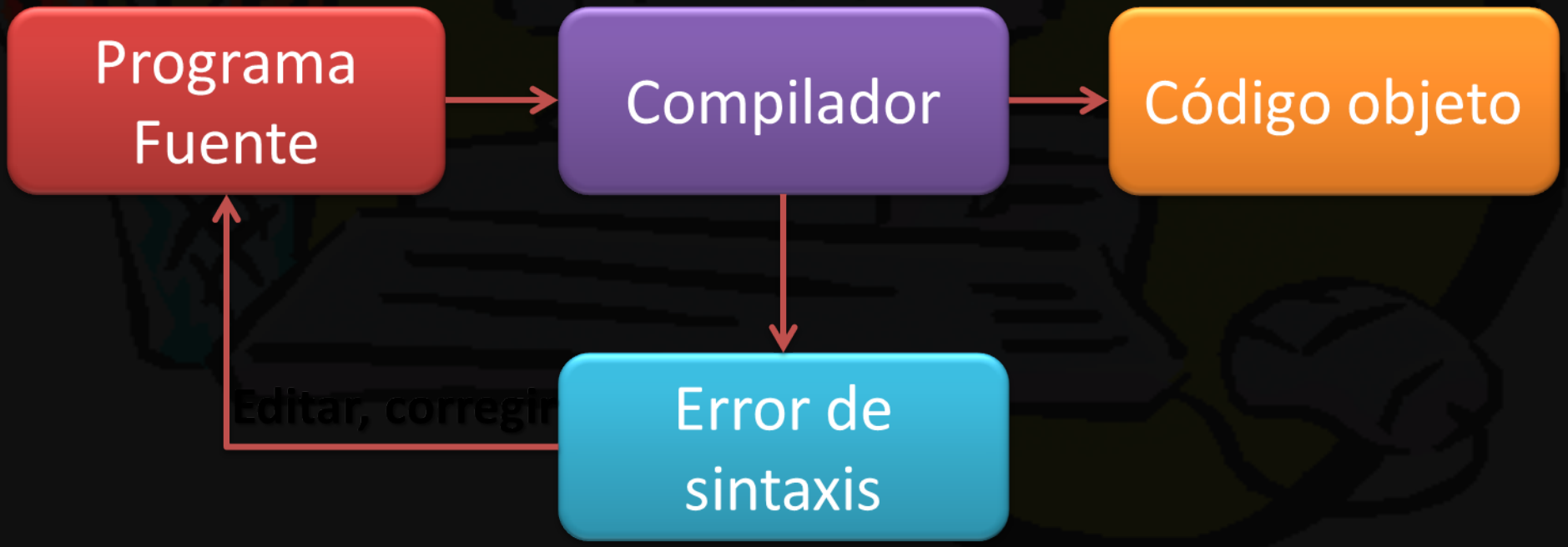
```c
#include<stdio.h>
main()
{
  int n;
  for(n=10;n>5;n--)
    printf("%d \t %d\n",n,n*n);
}
```

```
#include <stdio.h>

int numero;

int main(void)
{
	printf ("Introduzca un número entero: ");
	scanf ("%d", &numero);
	while (numero%2==0) {
		printf ("%d/2 = %d\n", numero, numero/2);
		numero/=2;
	}
	printf ("%d no es entero, saliendo...\n", numero);
	return 0;
}
```


Programa Fuente
Intérprete
Traduce y ejecuta línea a línea
Error de sintaxis
Editar, corregir


Programa Fuente
Compilador
Código objeto
Error de sintaxis
Editar, corregir

Bibliografia Basica:

http://www.tecnica1lomas.com.ar/tutoriales/lenguaje_C.pdf

“El Lenguaje de Porogramacion C”

Clasificación QA76.73C15 K46 1991

ISBN 968-880-205-0

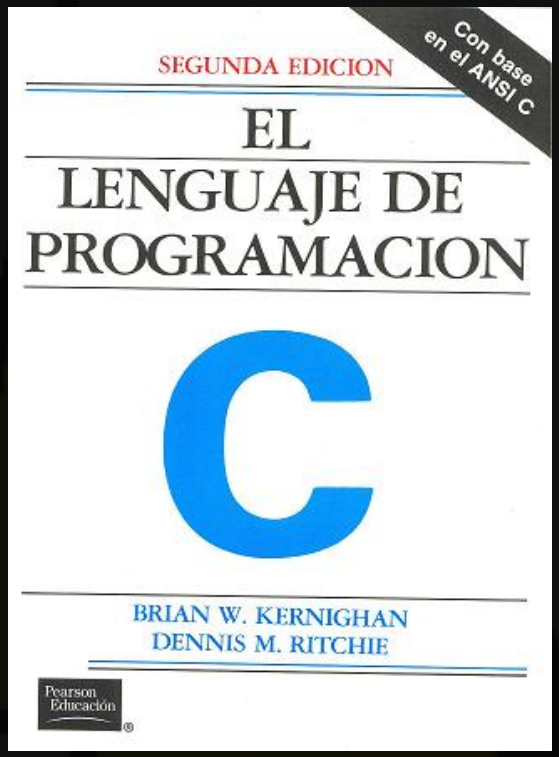
Autor Kernighan, Brian W. autor.

Título El lenguaje de programacion c / Brain w. kernighan, dennis m. ritchie ; tr. Néstor Gómez Muñoz

Edición 2

Datos de publicac. México : Prentice Hall, c1991

Descr. Física 294 páginas **Notas** Traduccion de: the c programming language

Materia C++ (Lenguaje de programación para computadora)

Sec. Personal Ritchie, Dennis M. Autor Gomez Muñoz, Nestor, traductor