

Herramientas Computacionales para Matemática Aplicada

Curso 2020

Introducción a la programación en C

Características del lenguaje de programación C

C es un lenguaje de programación de propósito general^{2:1} originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell,¹ como evolución del anterior lenguaje B, a su vez basado en BCPL.^{2:1 3 4}

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 sistema, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que 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.



The logo features a large, stylized blue letter 'C' with the words 'THE PROGRAMMING LANGUAGE' written in a smaller, blue, sans-serif font below it.

Desarrollador(es)	
Dennis Ritchie	y Laboratorios Bell
Información general	
Extensiones comunes	.c, .h
Paradigma	Imperativo (procedural), estructurado
Apareció en	1972
Diseñado por	Dennis Ritchie
Última versión estable	C18 (junio de 2018)
Sistema de tipos	Débil, estático
Implementaciones	GCC, Intel C, entre muchas más.
Dialectos	Cyclone, Unified Parallel C, Split-C, Cilk, C*
Influuido por	B (BCPL, CPL), ALGOL 68. ¹ 201-208
Ha influido a	Ensamblador, PL/I, Fortran Vala, C#, Objective-C, C++, AWK, bc, Java, JavaScript, PHP, Perl, NXC, D, Go

Características del lenguaje de programación C

Lenguajes derivados de C [editar]

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, pero actualmente usado en [Mac OS X](#), [iOS](#) y [GNUstep](#).
- [C++](#) (pronunciado *C Plus Plus*) 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 una sintaxis inspirada en la del C++ con 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](#).

En parte, a causa de ser de relativamente bajo nivel y de tener un modesto conjunto de características, se pueden desarrollar compiladores de C fácilmente. En consecuencia, el lenguaje C está disponible en un amplio abanico de plataformas (más que cualquier otro lenguaje). Además, a pesar de su naturaleza de bajo nivel, el lenguaje se desarrolló para incentivar la programación independiente de la máquina. Un programa escrito cumpliendo los estándares e intentando que sea portátil puede compilarse en muchos computadores.

C se desarrolló originalmente (conjuntamente con el sistema operativo [Unix](#), con el que ha estado asociado mucho tiempo) por programadores para programadores. Sin embargo, ha alcanzado una popularidad enorme, y se ha usado en contextos muy alejados de la [programación de software de sistema](#), para la que se diseñó originalmente.

Características del lenguaje de programación C

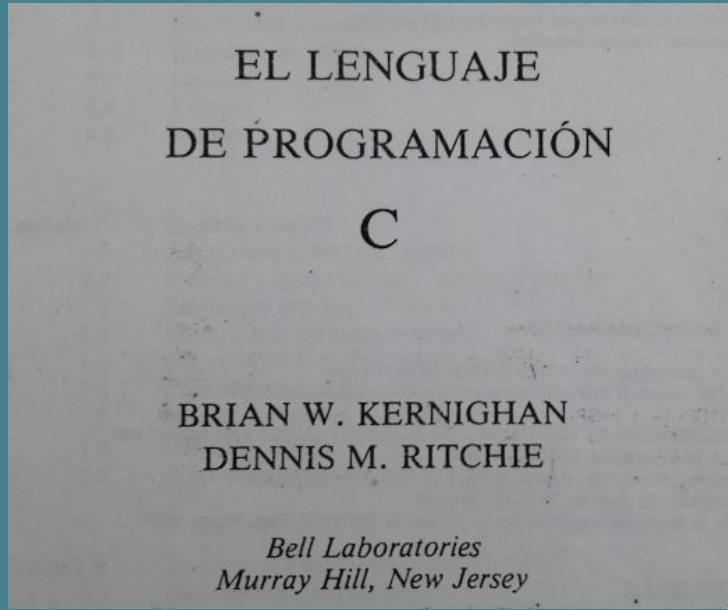
El C de Kernighan y Ritchie [editar]

En 1978, [Ritchie](#) y [Brian Kernighan](#) publicaron la primera edición de [El lenguaje de programación C](#), también conocido como [La biblia de C](#). Este libro fue durante años la especificación informal del lenguaje.^{2:2} El lenguaje descrito en este libro recibe habitualmente el nombre de "el C de Kernighan y Ritchie" o simplemente "K&R C" (La segunda edición del libro cubre el estándar [ANSI C](#), descrito más abajo).

Propiedades [editar]

- Núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de [gestión de archivos](#), proporcionadas por [bibliotecas](#).
- Es un lenguaje estructurado, i.e. tiene estructuras de control y tipos de datos estructurados definidos por el programador a partir de los tipos atómicos típicos y mediante arreglos, estructuras, uniones y apuntadores, incluidos los apuntadores a función.
- En su primera edición no había advertencias sobre asignar a una variable un valor un tipo distinto. Por lo que había un programa llamado [lint](#) que detectaba este tipo de errores. Actualmente los compiladores pueden detectar inconsistencias de tipos y otros errores.
- Usa un lenguaje de [preprocesado](#), el preprocesador de C, para tareas como definir [macros](#) e incluir múltiples archivos de [código fuente](#).
- Acceso a memoria de bajo nivel mediante el uso de [apuntadores o punteros](#).
- Manejo de Interrupciones mediante la biblioteca [signal](#).
- Un conjunto reducido de [palabras clave](#).
- El [llamado a funciones](#) es por valor. Aunque se pueden pasar apuntadores a variables para hacer llamados por referencia.
- Distintos tipos de almacenamiento que permiten un [diseño modular](#).

Fragmentos del manual de C de K&R (1978) ...



1.1 Comencemos

La única forma de aprender un nuevo lenguaje de programación es escribir programas en él. El primer programa por escribir es el mismo en todos los lenguajes:

8 EL LENGUAJE DE PROGRAMACIÓN C

CAPÍTULO I

Imprime las palabras.

hello, world

Este es el principal escollo. Para sobrepasarlo es necesario tener el texto del programa en algún sitio, compilarlo con éxito, cargarlo, ejecutarlo y averiguar dónde se quieren situar los resultados. Una vez superados estos detalles mecánicos, el resto es relativamente fácil.

En C, el programa para escribir "hello, world" (hola a todos) es

```
main()
{
    printf("hello, world\n");
}
```

La forma de ejecutar el programa depende del sistema concreto que se esté utilizando. He aquí un ejemplo específico: en el sistema operativo UNIX se debe crear el programa fuente en un archivo cuyo nombre acabe en ".c", como *hello.c*, y compilarlo con el comando.

cc hello.c

Si no se ha cometido ningún error (por ejemplo, omitir un carácter o cualquier otra equivocación), la compilación se realizará sin emitir mensajes de error, generando un archivo ejecutable llamado *a.out*. Su ejecución mediante el comando (orden)

a.out

producirá como salida

Hola, Mundo! en C ...

Creamos el código fuente



The screenshot shows a code editor window with the file "Hola.c" open. The code contains a single line of visible code: `printf("Hola, Mundo!\n"); //esta es la única línea de código que hace algo "visible" ;-)`.

```
#include <stdio.h>
/*Esto es un comentario*/
int main()
{
    printf("Hola, Mundo!\n"); //esta es la única línea de código que hace algo "visible" ;-)
    return(0);
}
```

At the bottom of the editor, the status bar displays: "C Tab Width: 8 Ln 11, Col 2 INS".

Compilamos y ejecutamos
el programa



The terminal window shows the following session:

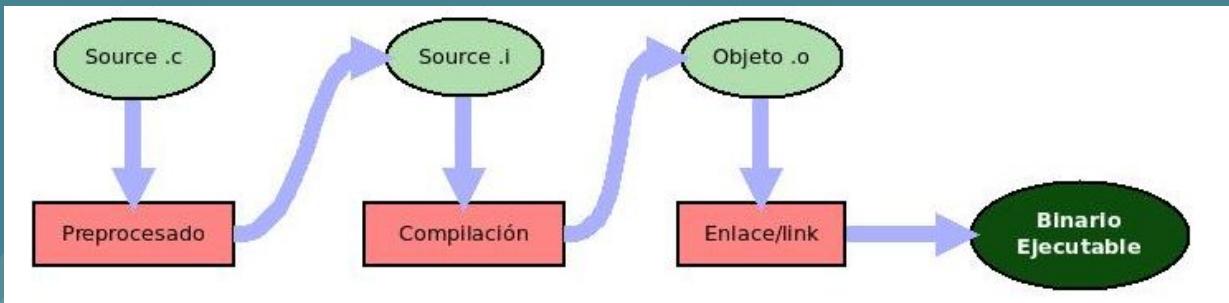
```
fedeLaptopLenovo:$ gcc Hola.c
fedeLaptopLenovo:$ ls -lrt a.out
-rwxr-xr-x 1 federico federico 8296 mar 30 11:26 a.out
fedeLaptopLenovo:$ ./a.out
Hola, Mundo!
fedeLaptopLenovo:$
```

El proceso de compilación en C

En pocas palabras, el proceso de compilación consiste en convertir uno o varios archivos de código fuente, en código binario ejecutable para una arquitectura de hardware/software determinada. En general, este proceso es dividido en varias etapas.

“Código fuente” es el programa que escribimos nosotros como programadores, el texto plano que le «dice» a la computadora cómo hacer las cosas.

Por otro lado, el código binario ejecutable, en general para cualquier lenguaje compilado, y en particular para lenguaje C, es código binario (no texto) que a su vez puede ser ejecutado en la computadora. Esta aclaración es necesaria, dado que uno de los productos intermedios del proceso de compilación es el código objeto, que si bien es binario, no puede ser ejecutado y debe continuar su proceso de compilación a la siguiente etapa, el enlace, o link.



Principales flags del compilador GNU de C gcc

Ejemplos:

- `gcc hola.c`
(compila el programa en C hola.c, genera un archivo ejecutable a.out)
- `gcc -o hola.x hola.c`
(compila el programa en C hola.c, genera un archivo ejecutable hola.x)
- `gcc -c hola.c`
(no genera el ejecutable, sino el código objeto, en el archivo hola.o)

Más opciones:

- E ~> realiza solamente el preprocessamiento, enviando el resultado a la salida estándar.
- I ruta ~> especifica la ruta hacia el directorio donde se encuentran los archivos marcados para incluir en el programa fuente. No lleva espacio entre la I y la ruta, así: -I/usr/include
- L ruta ~> especifica la ruta hacia el directorio donde se encuentran los archivos de biblioteca con el código objeto de las funciones referenciadas en el programa fuente. No lleva espacio entre la L y la ruta, así: -L/usr/lib
- Wall ~> muestra todos los mensajes de error y advertencia del compilador.
- g ~> incluye en el ejecutable generado la información necesaria para poder rastrear los errores usando un depurador, tal como GDB (GNU Debugger).

Ejemplo: hacer solo el preprocessado

Son habituales las siguientes extensiones o sufijos de los nombres de archivo:

.c	fuente en C
.C .cc .cpp .c++ .cp .cxx	fuente en C++; se recomienda .cpp
.m	fuente en Objective-C
.i	C preprocesado
.ii	C++ preprocesado
.s	fuente en lenguaje ensamblador
.o	código objeto
.h	archivo para preprocesador (encabezados), no suele figurar en la linea de comando de gcc

```
fedeLaptopLenovo:$ gcc -E Hola.c > Hola.i  
fedeLaptopLenovo:$ gedit Hola.i  
fedeLaptopLenovo:$
```



The screenshot shows a terminal window at the bottom left and a text editor window at the bottom right. The terminal window displays the command `gcc -E Hola.c > Hola.i` followed by `gedit Hola.i` and an empty line. The text editor window is titled "Hola.i" and shows the preprocessed code for the file "Hola.c". The code includes standard C library functions like `ftrylockfile` and `funlockfile` with specific attributes, and the main function definition.

```
extern int ftrylockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__));  
  
extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__));  
# 868 "/usr/include/stdio.h" 3 4  
  
# 2 "Hola.c" 2  
  
# 5 "Hola.c"  
int main()  
{  
  
printf("Hola, Mundo!\n");  
  
return(0);  
}
```

Tipos de datos y palabras reservadas en C

En 1983, el Instituto Nacional Estadounidense de Estándares (ANSI) organizó un comité, X3j11, para establecer una especificación estándar de C. Tras un proceso largo y arduo, se completó el estándar en 1989 y se ratificó como el "Lenguaje de Programación C" ANSI X3.159-1989. Esta versión del lenguaje se conoce a menudo como **ANSI C**, o a veces como C89 (para distinguirla de C99).

Palabras reservadas de ANSI-C [editar]

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

TIPOS DE DATOS BASICOS EN LENGUAJE C

TIPO	ANCHO EN BIT	RANGO EN PC
char	8	-128 a 127
int	16	-32768 a 32767
float	32	3.4E-38 a 3.4E+38
double	64	1.7E-308 a 1.7E+308
void	0	sin valores

TIPOS DE DATOS

TIPO	ANCHO EN BIT	RANGO EN PC
char	8	-128 a 127
unsigned char	8	0 a 255
signed char	8	-128 a 127
int	16	-32768 a 32767
unsigned int	16	0 a 65535
signed int	16	-32768 a 32767
short int	16	-32768 a 32767
unsigned short int	16	0 a 65535
signed short int	16	-32768 a 32767
long int	32	-2147483648 a 2147483647
signed long int	32	-2147483648 a 2147483647
unsigned long int	32	0 a 4294967295
float	32	3.4E-38 a 3.4E+38
double	64	1.7E-308 a 1.7E+308
long double	64	1.7E-308 a 1.7E+308

COMBINACIONES DE TIPOS DE DATOS

La biblioteca estándar de C

La **biblioteca estándar de C** (también conocida como **libc**) es una recopilación de ficheros cabecera y bibliotecas con rutinas, estandarizadas por un comité de la [Organización Internacional para la Estandarización](#) (ISO), que implementan operaciones comunes, tales como las de entrada y salida o el manejo de cadenas. A diferencia de otros lenguajes como [COBOL](#), [Fortran](#), o [PL/1](#), C no incluye palabras clave para estas tareas, por lo que prácticamente todo programa implementado en C se basa en la biblioteca estándar para funcionar.

El estándar ANSI [\[editar\]](#)

La biblioteca estándar de ANSI C consta de 24 ficheros cabecera que pueden ser incluidos en un proyecto de programación con una simple directiva. Cada cabecera contiene la declaración de una o más funciones, [tipos de datos y macros](#).

En comparación con otros lenguajes de programación (como por ejemplo [Java](#)) la biblioteca estándar es muy pequeña, ésta proporciona un conjunto básico de funciones matemáticas, de tratamiento de cadenas, conversiones de tipo y entrada/salida por [consola](#) o por ficheros. No se incluyen, ni un conjunto de tipos de datos contenedores básicos (listas, pilas, colas, ...), ni herramientas para crear una [interfaz gráfica de usuario](#) (GUI), ni operaciones para trabajar en red, ni otras funcionalidades que lenguajes como C++ o Java incorporan de manera estándar. La principal ventaja del reducido tamaño de la biblioteca estándar de C es que construir un entorno de trabajo en ANSI C es muy fácil y, en consecuencia, portar un programa en ANSI C de una plataforma a otra es relativamente sencillo.

Se han desarrollado muchas otras bibliotecas para proporcionar una funcionalidad equivalente a la de otros lenguajes de programación. Por ejemplo, el proyecto de desarrollo del entorno de escritorio de GNOME creó las bibliotecas GTK+ y GLib con funcionalidades para desarrollar y trabajar con interfaces gráficas de usuario

Algunos de los archivos de cabecera más importantes

<math.h>	Contiene las funciones matemáticas comunes.
<stdio.h>	Proporciona el núcleo de las capacidades de entrada/salida del lenguaje C (incluye la venerable función <code>printf</code>).
<stdlib.h>	Para realizar ciertas operaciones como conversión de tipos, generación de números pseudo-aleatorios, gestión de memoria dinámica, control de procesos, funciones de entorno, de señalización (??), de ordenación y búsqueda.
<string.h>	Para manipulación de cadenas de caracteres .
<time.h>	Para tratamiento y conversión entre formatos de fecha y hora.
<float.h>	Contiene la definición de constantes que especifican ciertas propiedades de la biblioteca de coma flotante, como la diferencia mínima entre dos números en coma flotante (<code>_EPSILON</code>), el número máximo de dígitos de precisión (<code>_DIG</code>), o el rango de valores que se pueden representar (<code>_MIN, _MAX</code>).

Ciclos while

```
ciclos_while.c  x
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int k;
6     int N=20;
7     int suma=0;
8
9     k=1;
10    while(k <= N){
11        suma+=k;
12        k++;
13    }
14
15 /*imprimimos la salida*/
16 printf("La suma de los primeros %d naturales es %d\n", N, suma);
17
18 printf("También se puede hacer directamente con la fórmula N(N+1)/2 = %d\n", N*(N+1)/2);
19
20 return(0);
21 }
22
```

```
fedelaptopLenovo:$ gcc ciclos_while.c -o ciclos_while -lm
fedelaptopLenovo:$ ./ciclos_while
La suma de los primeros 20 naturales es 210
También se puede hacer directamente con la fórmula N(N+1)/2 = 210
```

Ciclos for

```
ciclos_for.c x
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(void)
5 {
6     int k;
7     int N=20;
8     int suma_cuad=0;
9
10    for(k=1; k<=N; k++){
11        suma_cuad+=pow(k,2);
12    }
13
14    /*imprimimos la salida*/
15    printf("La suma de los primeros %d naturales al cuadrado es %d\n", N, suma_cuad);
16
17    printf("También se puede hacer directamente con la fórmula N(N+1)(2N+1)/6 = %d\n",
18           N*(N+1)*(2*N+1)/6);
19
20    return(0);
21 }
22
```

```
fedeLaptopLenovo:$ ./ciclos_for
La suma de los primeros 20 naturales al cuadrado es 2870
También se puede hacer directamente con la fórmula N(N+1)(2N+1)/6 = 2870
fedeLaptopLenovo:$
```

Dónde está el error?

```
aritmetica_int_vs_double.c x
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int k;
6     double x;
7     double dif_mal,dif_bien;
8     /*asignamos valor 12 a k y copiamos ese valor a x*/
9     k = 12;
10    x = k;
11
12    /*12/7 devuelve 1! es una división entre enteros*/
13    dif_mal = x/7-k/7;
14    /*con
15     dif_bien = x/7.0-k/7.0;
16     alcanza, pero mejor hacer un cast explícito,
17     para que no queden dudas...
18 */
19    dif_bien = x/7.0-(double)k/7.0;
20
21    /*imprimimos la salida...(extraña, no?)*/
22    printf("dif_mal = %.3f\n", dif_mal);
23    printf("dif_bien = %.3f\n", dif_bien);
24
25    return(0);
26 }
```

```
fedeLaptopLenovo:$ gcc aritmetica_int_vs_double.c
fedeLaptopLenovo:$ ./a.out
dif_mal = 0.714
dif_bien = 0.000
```

Cuidado! No es lo mismo la aritmética entre enteros (int) que entre variables en punto flotante (float o double)!!!

Códigos de formato de printf/scanf

Código	Formato
%c	un char (caracter)
%d	un entero con signo en notación de base decimal
%i	un entero con signo
%e	reales((pseudoreales como double)) en notación científica indicando el exponente con "e"
%E	reales((pseudoreales como double)) en notación científica indicando el exponente con "E"
%f	formato de punto flotante
%g	la opción más corta entre "%e" y "%f"
%G	la opción más corta entre "%E" y "%F"
%o	un entero sin signo en notación de base octal
%s	una cadena de caracteres
%u	un entero sin signo
%x	un entero sin signo en notación de base hexadecimal, usando minúsculas para los dígitos extendidos
%X	un entero sin signo en notación de base hexadecimal, usando mayúsculas para los dígitos extendidos
%p	un puntero
%n	un puntero a un entero en el cual se deposita la cantidad de caracteres escritos hasta el momento

Formateador	Salida
%07i	justificado a la derecha, 7 dígitos de largo, sin relleno
%.7i	largo mínimo de 7 dígitos, justificado a la derecha, rellena con ceros
%8.2f	tamaño total de 8 dígitos, con dos decimales

Entrada de datos con scanf

```
ej_scanf.c      x
1 #include <stdio.h>
2
3
4 int main(void)
5 {
6     int aux;
7
8     printf("Por favor, introduce un entero múltiplo de 7\n");
9
10    scanf("%d", &aux);
11
12    if( (aux%7) == 0 ){
13        printf("Bien hecho!\n%d es múltiplo de 7\n\n", aux);
14    }
15    else{
16        printf("Ooops... \n%d NO es múltiplo de 7...\n\n", aux);
17    }
18
19    return(0);
20 }
21
```

```
fedeLaptopLenovo:$ gcc ej_scanf.c
fedeLaptopLenovo:$ ./a.out
Por favor, introduce un entero múltiplo de 7
34
Ooops...
34 NO es múltiplo de 7...
```

```
fedeLaptopLenovo:$ ./a.out
Por favor, introduce un entero múltiplo de 7
49
Bien hecho!
49 es múltiplo de 7
```

Estructuras condicionales if - else if

```
if_else.c      x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int num = 14;
7
8     if( num%4 == 0 ){
9         printf("%d es divisible por 4\n", num);
10    }
11    else if( num%4 == 1 ){
12        printf("El resto de la división de %d por 4 es %d\n", num, 1);
13    }
14    else if( num%4 == 2 ){
15        printf("El resto de la división de %d por 4 es %d\n", num, 2);
16    }
17    else if( num%4 == 3 ){
18        printf("El resto de la división de %d por 4 es %d\n", num, 3);
19    }
20    else{
21        printf("Este caso no se tendría que haber presentado...se cancela la ejecución.\n");
22        exit(0);
23    }
24
25 return(0);
26 }
```

```
fedeLaptopLenovo:$ gcc if_else.c -lm
fedeLaptopLenovo:$ ./a.out
El resto de la división de 14 por 4 es 2
```

Estructuras switch - case

```
1 #include<stdio.h>
2
3 int main()
4 {
5     char ch;
6
7     printf("Introduzca una vocal en minúscula: ");
8
9     ch=getchar();
10
11    switch(ch) {
12        case 'a':
13            printf("Se ha pulsado una a.\n");
14            break;
15        case 'e':
16            printf("Se ha pulsado una e.\n");
17            break;
18        case 'i':
19            printf("Se ha pulsado una i.\n");
20            break;
21        case 'o':
22            printf("Se ha pulsado una o.\n");
23            break;
24        case 'u':
25            printf("Se ha pulsado una u.\n");
26            break;
27        default:
28            printf("Error: No se pulsó una vocal en minúscula...\n");
29        }
30    return(1);
31 }
32 }
```

```
fedeLaptopLenovo:$ gcc switch.c -lm
fedeLaptopLenovo:$ ./a.out
Introduzca una vocal en minúscula: a
Se ha pulsado una a.
fedeLaptopLenovo:$ ./a.out
Introduzca una vocal en minúscula: r
Error: No se pulsó una vocal en minúscula...
fedeLaptopLenovo:$ ./a.out
Introduzca una vocal en minúscula: E
Error: No se pulsó una vocal en minúscula...
fedeLaptopLenovo:$ ./a.out
Introduzca una vocal en minúscula: u
Se ha pulsado una u.
fedeLaptopLenovo:$ █
```

Definición de funciones

```
funciones.c      x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 double pot_cubo(double x)
5 {
6     return(x*x*x);
7 }
8
9 void imprimir_real(double x)
10 {
11     printf("%.5g\n", x);
12 }
13
14 int main(void)
15 {
16     int k;
17     double x;
18
19     x=pot_cubo(2.2);
20
21     imprimir_real(x);
22
23     for (k = 0; k < 10; k++)
24     {
25         printf("%d\t%.3g\n", k, pot_cubo(k));
26     }
27
28     return(0);
29 }
30
```

```
$gcc funciones.c -lm
$./a.out
10.648
0      0
1      1
2      8
3      27
4      64
5      125
6      216
7      343
8      512
9      729
```

Arreglos multidimensionales

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 double media_arr(double *arr, int dim)
5 {
6     double ret=0.0;
7     int i;
8
9     for (i = 0; i < dim; ++i)
10    {
11        ret+=arr[i];
12    }
13
14    ret=ret/(double)dim;
15
16    return(ret);
17}
18
```

```
$gcc arreglos.c
$./a.out
1.20      0.00
0.50      2.30
7.5
```

```
19 int main(void)
20 {
21     int i,k;
22     double Vect[]={5.0, 6.0, 8.3, 5.7, 12.5};
23     double Mat[2][2];
24
25     Mat[0][0]=1.2;
26     Mat[0][1]=0.0;
27     Mat[1][0]=0.5;
28     Mat[1][1]=2.3;
29
30     for (i = 0; i < 2; i++)
31    {
32         printf("\n");
33         for (k = 0; k < 2; k++)
34        {
35             printf("%.2f\t", Mat[i][k]);
36         }
37     }
38
39
40     printf("\n\n%.3g\n", media_arr(Vect, 5));
41
42
43
44     return(0);
45}
46
```

Pasando argumentos desde la línea de comandos

```
pulg2cm.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /*compilar con:
5 gcc pulg2cm.c -o pulg2cm -lm -O3
6 */
7 char usage[]="./pulg2cm long_en_pulgadas\n";
8
9 int num_args=2;
10
11 char descripcion[]="El programa recibe \
12 desde línea de comandos un número que representa una longitud en pulgadas y la transforma a centímetros.\n";
13
14 char error_numargs[]="Número incorrecto de argumentos.\n";
15
16
17 int main(int numarg,char **cargs)
18 {
19     double val_pulg, val_cm;
20     printf("%s",descripción);
21
22     if(numarg != num_args){
23         printf("%s",error_numargs);
24         printf("%s",usage);
25         exit(1);
26     }
27
28     val_pulg=atof(cargs[1]); /*leemos el argumento pasado*/
29
30
31     val_cm=2.54*val_pulg; /*hacemos la conversión*/
32
33     /*imprimimos la salida*/
34     printf("%.3f pulgadas equivalen a %.3f centímetros\n", val_pulg, val_cm);
35
36     return(0);
37 }
```

```
fedeLaptopLenovo:$ gcc pulg2cm.c -o pulg2cm -lm -O3
fedeLaptopLenovo:$ ./pulg2cm 12
```

El programa recibe desde línea de comandos un número que representa una longitud en pulgadas y la transforma a centímetros.
12.000 pulgadas equivalen a 30.480 centímetros

Funciones matemáticas y constantes definidas en math.h

acos	arcocoseno
asin	arcoseno
atan	arcotangente
atan2	arcotangente de dos parámetros
floor	función suelo
cos	coseno
cosh	coseno hiperbólico
exp(double x)	función exponencial, computa e^x
fabs	valor entero
ceil	menor entero no menor que el parámetro
fmod	residuo de la división de flotantes

log	logaritmo natural
log10	logaritmo en base 10
modf	obtiene un valor en punto flotante íntegro y en partes
pow	eleva un valor dado a un exponente, x^y
sin	seno
sinh	seno hiperbólico
sqrt	raíz cuadrada
tan	tangente
tanh	tangente hiperbólica

Name	Description
M_E	The base of natural logarithms.
M_LOG2E	The logarithm to base 2 of M_E.
M_LOG10E	The logarithm to base 10 of M_E.
M_LN2	The natural logarithm of 2.
M_LN10	The natural logarithm of 10.
M_PI	Pi, the ratio of a circle's circumference to its diameter.
M_PI_2	Pi divided by two.
M_PI_4	Pi divided by four.
M_1_PI	The reciprocal of pi (1/pi).
M_2_PI	Two times the reciprocal of pi.
M_2_SQRTPI	Two times the reciprocal of the square root of pi.
M_SQRT2	The square root of two.
M_SQRT1_2	The reciprocal of the square root of two (also the square root of 1/2).

Funciones matemáticas y constantes definidas en math.h (C99)

C99 functions [\[edit\]](#)

Name	Description
acosh	inverse hyperbolic cosine
asinh	inverse hyperbolic sine
atanh	inverse hyperbolic tangent
cbrt	cube root
copysign(x,y)	returns the value of x with the sign of y
erf	error function
erfc	complementary error function
exp2(x)	raise 2 to the power of x , 2^x
expm1(x)	one less than the exponential of x , $e^x - 1$
fdim(x,y)	positive difference between x and y , $\max(x-y, 0)$
fma(x,y,z)	multiply and add, $(x * y) + z$
fmax(x,y)	largest value of x and y
fmin(x,y)	smallest value of x and y
hypot(x,y)	hypotenuse, $\sqrt{x^2 + y^2}$
ilogb	the exponent of a floating-point value, converted to an <code>int</code>
lgamma	natural log of the absolute value of the gamma function
llrint	round to integer (returns <code>long long</code>) using current rounding mode

llround	round to integer (returns <code>long long</code>)
lround	round to integer (returns <code>long</code>)
log1p(x)	natural logarithm of $1 + x$
log2	base-2 logarithm
logb	extract exponent from floating-point number
nan(s)	returns NaN, possibly using string argument
nearbyint	round floating-point number to nearest integer
nextafter(x,y)	returns next representable value after x (towards y)
nexttoward(x,y)	same as <code>nextafter</code> , except y is always a <code>long double</code>
remainder(x,y)	calculates remainder, as required by IEC 60559
remquo(x,y,p)	same as <code>remainder</code> , but store quotient (as <code>int</code>) at target of pointer p
rint	round to integer (returns <code>double</code>) using current rounding mode
round	round to integer (returns <code>double</code>), rounding halfway cases away from zero
scalbln(x,n)	$x * \text{FLT_RADIX}^n$ (n is <code>long</code>)
scalbn(x,n)	$x * \text{FLT_RADIX}^n$ (n is <code>int</code>)
tgamma	gamma function
trunc	truncate floating-point number

Llamadas al sistema

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6 /*imprimimos la fecha y hora actual mediante una llamada al
7 comando del sistema date*/
8 system("date");
9
10 printf("\n\n");
11
12 /*imprimimos el calendario del mes en curso mediante el
13 comando del sistema cal*/
14 system("cal 04 2020");
15
16 return(0);
17 }
18
```

```
fedeLaptopLenovo:$ gcc sistema.c -lm
fedeLaptopLenovo:$ ./a.out
lun mar 30 15:50:03 -03 2020
```

Abril 2020						
do	lu	ma	mi	ju	vi	sá
				1	2	3
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Generación de números aleatorios

```
num_aleatorios.c  x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int semilla = 12345;
7     double x;
8
9     /*inicializamos el generador de números aleatorios*/
10    srand(semilla);
11
12    /*x será una variable aleatoria distribuida de manera uniforme en [0,1]*/
13    x=rand()/(RAND_MAX+1.0);
14
15    /*simulamos la tirada de una moneda*/
16    if(x<0.5){
17        printf("La moneda salió cara\n");
18    }
19    else{
20        printf("La moneda salió seca\n");
21    }
22
23    return(0);
24
25
```

```
fedeLaptopLenovo:$ gcc num_aleatorios.c -o num_aleatorios -lm
fedeLaptopLenovo:$ ./num_aleatorios
La moneda salió cara
```

Generación de números aleatorios con semillas aleatorias

```
num_aleatorios_1.c •
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 int main(void)
6 {
7     int semilla;
8     double x;
9
10 /*
11  * invocamos a la función time(0) (número de segundos transcurridos desde 00:00:00 UTC, January 1, 1970)
12  * para generar una semilla distinta para el generador
13  * de números aleatorios, de manera que en distintas ejecuciones del programa se obtengan
14  * secuencias distintas de números aleatorios
15 */
16
17 semilla = time(0);
18 /*inicializamos el generador de números aleatorios*/
19 srand(semilla);
20
21 /*x será una variable aleatoria distribuida de manera uniforme en [0,1]*/
22 x=rand()/(RAND_MAX+1.0);
23
24 /*simulamos la tirada de una moneda*/
25 if(x<0.5){
26     printf("La moneda salió cara\n");
27 }
28 else{
29     printf("La moneda salió seca\n");
30 }
31
32 return(0);
33 }
```

```
fedeLaptopLenovo:$ gcc num_aleatorios_1.c -o num_aleatorios_1 -lm
fedeLaptopLenovo:$ ./num_aleatorios_1
La moneda salió cara
fedeLaptopLenovo:$ ./num_aleatorios_1
La moneda salió cara
fedeLaptopLenovo:$ ./num_aleatorios_1
La moneda salió seca
fedeLaptopLenovo:$ ./num_aleatorios_1
La moneda salió seca
fedeLaptopLenovo:$ ./num_aleatorios_1
La moneda salió cara
```

Estructuras

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 typedef struct NumComplejo{
6     double Re;
7     double Im;
8     } complejo;
9
10
11 complejo Conjugado(complejo a){
12     complejo ret ={a.Re,-a.Im};
13     return(ret);
14 }
15
16 complejo Suma(complejo a, complejo b){
17     complejo ret ={a.Re + b.Re, a.Im + b.Im};
18     return(ret);
19 }
20
21 void ImprimirComplejo(complejo a)
22 {
23     printf("( %.3f , %.3f )\n", a.Re, a.Im);
24 }
25
```

```
$gcc estructuras.c
$./a.out
z = ( 2.000 , 3.000 )
z1 = ( 4.000 , 0.000 )
w = ( 2.000 , -3.000 )
w1 = ( 4.000 , -6.000 )
```

```
26
27 int main(void)
28 {
29     complejo z, z1, w, w1;
30
31     z.Re = 2.0;
32     z.Im = 3.0;
33
34     w = Conjugado(z);
35
36     z1 = Suma(w, z);
37
38     w1 = Conjugado(Suma(z, z));
39
40     printf("z = ");
41     ImprimirComplejo(z);
42     printf("z1 = ");
43     ImprimirComplejo(z1);
44     printf("w = ");
45     ImprimirComplejo(w);
46     printf("w1 = ");
47     ImprimirComplejo(w1);
48
49     return(0);
50 }
51
```

Procesando señales

```
signal.c      x
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <signal.h>
5
6 /*Este programa es solo un ejemplo para aprender a manejar seniales.
7 el programa cuenta la serie natural (un numero por segundo), pero cuando se
8 recibe una senial ctrl-z, interrumpe la cuenta e imprime un mensaje, luego sigue
9 la cuenta
10 */
11 void toma_señal(int sig_num)
12 {
13     signal(SIGTSTP,toma_señal);
14     printf("Se ha recibido una señal\n");
15     fflush(stdout);
16 }
17
18 int main (void)
19 {
20     int k=0;
21     signal(SIGTSTP,toma_señal);
22     while (1){
23         printf("%d\n",k);
24         sleep(1);
25         k++;
26     }
27
28     return(0);
29 }
30
```

```
$gcc signal.c
$./a.out
0
1
2
3
^ZSe ha recibido una señal
4
5
6
7
^ZSe ha recibido una señal
8
9
10
11
^ZSe ha recibido una señal
12
```

Arreglos y apuntadores

```
punteros.c      x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 double media_arr(double *arr, int dim)
5 {
6     double ret=0.0;
7     int i;
8
9     for (i = 0; i < dim; ++i)
10    {
11        ret+=arr[i];
12    }
13
14 ret=ret/(double)dim;
15
16 return(ret);
17 }
```

```
19 int main(void)
20 {
21     double Vect[]={5.0, 6.0, 8.3, 5.7, 12.5};
22     double *pt; /*declaramos el puntero*/
23     double x = 32.0;
24     double y;
25
26     pt = &x; /*pt "apunta" a la dirección de x. Esto se
27                 | denominá "referenciar" al puntero*/
28
29 /*ahora "desreferenciamos" el puntero*/
30     y = *pt;
31
32     printf("%.3f\n", y);
33
34     printf("%.3f\n", *pt);
35
36     pt = Vect; /*ahora pt apunta a la misma dirección
37                 | de memoria que el arrglo Vect*/
38
39     printf("\n%.3g\n", media_arr(Vect, 5));
40     printf("\n%.3g\n", media_arr(pt, 5));
41
42     printf("\n%.3g\n", Vect[3]);
43     printf("\n%.3g\n", *(Vect+3));
44     printf("\n%.3g\n", *(pt+3));
45
46
47
48     return(0);
49 }
```

```
$gcc punteros.c
$./a.out
32.000
32.000
7.5
7.5
5.7
5.7
5.7
```

Asignación dinámica de memoria

```
◀ ▶ din_alloc.c      x
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 char usage[] = "./din_alloc N\n";
6
7 int num_args = 2;
8
9
10 char error_numargs[] = "Número incorrecto de argumentos.\n";
11
12
13 int main(int numarg, char **cargs)
14 {
15     double *XDin;
16     double XStat[20]; /*reserva estática de memoria*/
17
18     if (numarg != num_args){
19         printf("%s", error_numargs);
20         printf("%s", usage);
21         exit(1);
22     }
23
24 /*reserva dinámica de memoria*/
25 XDin = (double *)malloc(atoi(cargs[1]) * sizeof(double));
26
```

Asignación dinámica de memoria (cont.)

```
26
27 /*asignación en arreglo estático*/
28 for (int i = 0; i < 20; ++i)
29 {
30     XStat[i] = pow(i,0.5);
31 }
32
33 /*asignación en arreglo dinámico*/
34 for (int i = 0; i < atoi(cargs[1]); ++i)
35 {
36     XDin[i] = pow(i,2.0);
37 }
38
39
40 printf("\n%s\n", "Elementos del arreglo estático:");
41 for (int i = 0; i < 20; ++i)
42 {
43     printf("%.3g;\t", XStat[i]);
44 }
45
46 printf("\n%s\n", "Elementos del arreglo dinámico:");
47 for (int i = 0; i < atoi(cargs[1]); ++i)
48 {
49     printf("%.3g;\t", XDin[i]);
50 }
51
52 printf("\n");
53
54 free(XDin); /*liberamos la memoria solicitada*/
55
56 return(0);
57 }
```

```
$gcc din_alloc.c -o din_alloc -lm
$./din_alloc 15
```

Elementos del arreglo estático:

0; 1; 1.41; 1.73; 2; 2.24; 2.45;
2.65; 2.83; 3; 3.16; 3.32; 3.46; 3.61;
3.74; 3.87; 4; 4.12; 4.24; 4.36;

Elementos del arreglo dinámico:

0; 1; 4; 9; 16; 25; 36;
49; 64; 81; 100; 121; 144; 169;
196;

Lectura-escritura de archivos

```
file_in_out.c      x
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 int main(void)
6 {
7     FILE *fopen(), *fin, *fout;
8     float Data[10];
9     float aux;
10
11
12    /*leemos desde el archivo de entrada*/
13    fin=fopen("DatosIn.dat","r");
14    for (int i = 0; i < 10; ++i)
15    {
16        fscanf(fin, "%f", &aux);
17        Data[i] = aux;
18    }
19
20
21    /*imprimimos en el archivo de salida*/
22    fout=fopen("DatosOut.dat","w");
23    for (int i = 0; i < 10; ++i)
24    {
25        fprintf(fout, "%3.3g\n", pow(Data[i],2));
26    }
27
28
29    fclose(fin);
30    fclose(fout);
31
32    return(0);
33 }
```

DatosIn.dat
~/tmp/c_code

```
0.1
1.1
2.2
3.3
4.0
5.2
6.6
7.1
8.0
9.3 |
```

ext ▾ Tab Width: 8 ▾ Ln 10, Col 5 ▾ INS

DatosOut....
~/tmp/c_code

```
0.01
1.21
4.84
10.9
16
27
43.6
50.4
64
86.5 |
```

ext ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

Algunas de las funciones para manejo de cadenas en string.h

<code>strcat</code>	añade una cadena al final de otra
<code>strncat</code>	añade los n primeros caracteres de una cadena al final de otra
<code>strchr</code>	localiza un carácter en una cadena, buscando desde el principio
<code> strrchr</code>	localiza un carácter en una cadena, buscando desde el final
<code>strcmp</code>	compara dos cadenas alfabéticamente ('a'!= 'A')
<code>strncmp</code>	compara los n primeros caracteres de dos cadenas numéricamente ('a'!= 'A')
<code>strcoll</code>	compara dos cadenas según la colación actual ('a'== 'A')
<code>strcpy</code>	copia una cadena en otra
<code>strncpy</code>	copia los n primeros caracteres de una cadena en otra
<code>strerror</code>	devuelve la cadena con el mensaje de error correspondiente al número de error dado
<code>strlen</code>	devuelve la longitud de una cadena
<code>strspn</code>	devuelve la posición del primer carácter de una cadena que no coincide con ninguno de los caracteres de otra cadena dada
<code>strcspn</code>	devuelve la posición del primer carácter que coincide con alguno de los caracteres de otra cadena dada
<code>strupr</code>	encuentra la primera ocurrencia de alguno de los caracteres de una cadena dada en otra
<code>strstr</code>	busca una cadena dentro de otra
<code>strtok</code>	parte una cadena en una secuencia de tokens
<code>strxfrm</code>	transforma una cadena en su forma de colación (??)
<code>strrev</code>	invierte una cadena

Ejemplo de manipulación de cadenas de caracteres

```
cadenas.c      x
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5
6 int main(void)
7 {
8     char pregunta[] = "Cual es el apellido del creador de la Teoría de la Relatividad?";
9     char input[50];
10    char Respuesta[] = "Einstein";
11    char Nombre[] = "Albert ";
12
13    printf("%s\n", pregunta);
14
15    scanf("%s", input);
16
17    printf("%s%s\n", "Respuesta leída: ", input);
18
19
20    if (strcasecmp(input, Respuesta) == 0){
21        printf("%s%s\n", "Muy bien!, el creador de la Teoría de la Relatividad fue ",
22               strncat(Nombre, Respuesta));
23    }
24
25    else{
26        printf("%s%s\n", "No no..., el creador de la Teoría de la Relatividad fue ",
27               strncat(Nombre, Respuesta));
28    }
29
30
31    return(0);
32 }
```

Ejemplo de manipulación de cadenas de caracteres

```
$gcc cadenas.c -lm
```

```
$./a.out
```

Cual es el apellido del creador de la Teoría de la Relatividad?

```
newton
```

Respuesta leída: newton

No no..., el creador de la Teoría de la Relatividad fue Albert Einstein

```
$./a.out
```

Cual es el apellido del creador de la Teoría de la Relatividad?

```
Einstein
```

Respuesta leída: Einstein

Muy bien!, el creador de la Teoría de la Relatividad fue Albert Einstein

```
$./a.out
```

Cual es el apellido del creador de la Teoría de la Relatividad?

```
EinsTeIN
```

Respuesta leída: EinsTeIN

Muy bien!, el creador de la Teoría de la Relatividad fue Albert Einstein

Ejemplo de creación de Makefiles

~/mfc_cma

```
File Edit Selection Find View Goto Tools Project  
cubo.c lib_cubo.c  
1 #include <math.h>  
2  
3 double cubo(double x)  
4 {  
5     return(pow(x,3));  
6 }  
7
```

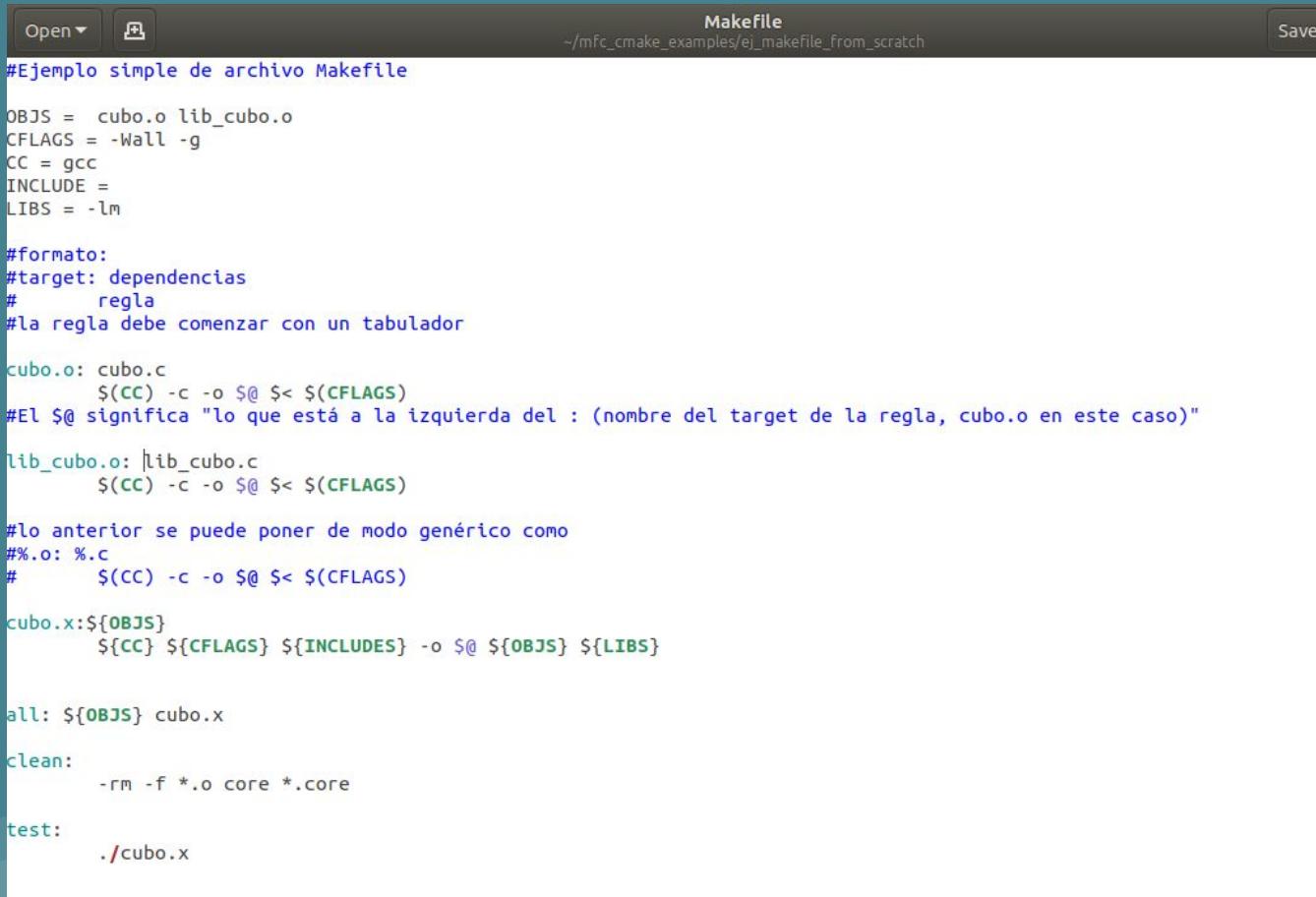
lib_cubo.c contiene función a ser llamada en main

cubo.c contiene la función main

~/mrc_cmake_examples/ej_makerror

```
File Edit Selection Find View Goto Tools Preferences Help  
cubo.c  
1 #include <stdio.h>  
2  
3 double cubo(double);  
4  
5 int main()  
6 {  
7     double x=4.2;  
8  
9     printf("El valor de %.4f al cubo es: %.4f\n", x, cubo(x));  
10  
11    return(0);  
12 }  
13
```

Makefile para este ejemplo:



The screenshot shows a code editor window titled "Makefile" with the file path "~/mfc_cmake_examples/ej_makefile_from_scratch". The interface includes standard buttons for "Open", "Save", and "Find". The code itself is a Makefile with the following content:

```
#Ejemplo simple de archivo Makefile

OBJS = cubo.o lib_cubo.o
CFLAGS = -Wall -g
CC = gcc
INCLUDE =
LIBS = -lm

#formato:
#target: dependencias
#      regla
#la regla debe comenzar con un tabulador

cubo.o: cubo.c
    $(CC) -c -o $@ $< $(CFLAGS)
#El $@ significa "lo que está a la izquierda del : (nombre del target de la regla, cubo.o en este caso)"

lib_cubo.o: lib_cubo.c
    $(CC) -c -o $@ $< $(CFLAGS)

#lo anterior se puede poner de modo genérico como
%.o: %.c
#      $(CC) -c -o $@ $< $(CFLAGS)

cubo.x:${OBJS}
    $(CC) ${CFLAGS} ${INCLUDES} -o $@ ${OBJS} ${LIBS}

all: ${OBJS} cubo.x

clean:
    -rm -f *.o core *.core

test:
    ./cubo.x
```

Invocando make:

```
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ ls  
cubo.c  cubo.x  lib_cubo.c  Makefile  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ make cubo.o  
gcc -c -o cubo.o cubo.c -Wall -g  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ make all  
gcc -c -o lib_cubo.o lib_cubo.c -Wall -g  
gcc -Wall -g -o cubo.x cubo.o lib_cubo.o -lm  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ make test  
.cubo.x  
El valor de 4.2000 al cubo es: 74.0880  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ ls  
cubo.c  cubo.o  cubo.x  lib_cubo.c  lib_cubo.o  Makefile  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ make clean  
rm -f *.o core *.core  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ ls  
cubo.c  cubo.x  lib_cubo.c  Makefile  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ make cubo.x  
gcc -c -o cubo.o cubo.c -Wall -g  
gcc -c -o lib_cubo.o lib_cubo.c -Wall -g  
gcc -Wall -g -o cubo.x cubo.o lib_cubo.o -lm  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$ ls  
cubo.c  cubo.o  cubo.x  lib_cubo.c  lib_cubo.o  Makefile  
fede@fedeLaptopLenovo: ~/mfc_cmake_examples/ej_makefile_from_scratch$
```

Programas con salida gráfica con la biblioteca GTK

GTK

En informática, **GTK** (conocido hasta febrero de 2019 como **GTK+**^{1 2}) o *The GIMP Toolkit*^{3 4:3} es una [biblioteca](#) de componentes gráficos [multiplataforma](#) para desarrollar [interfaces gráficas de usuario \(GUI\)](#).^{5 4:3} Fue desarrollada inicialmente para implementar la interfaz gráfica del programa de edición de imágenes **GIMP**.^{3 4:3} En 1997 el proyecto **GNOME** escogió GTK+ como base sobre la cual desarrollar su [entorno gráfico](#).⁶ Otras aplicaciones gráficas no directamente relacionadas con GNOME también han empleado GTK, convirtiéndose junto con **Qt** en la biblioteca GUI más popular del sistema operativo **Linux**.^[cita requerida] Otros entornos gráficos para Linux como **Xfce** y **ROX** han elegido también GTK como su biblioteca de componentes gráficos.^[cita requerida] GTK también se puede emplear para desarrollar aplicaciones gráficas que funcionen en los escritorios de **Microsoft Windows**, **Mac OS** y otros sistemas operativos.^[cita requerida]

Licenciado bajo los términos de **GNU LGPL**,⁵ GTK permite la creación de tanto [software libre](#) como [software privativo](#).^[cita requerida] GTK es parte del proyecto **GNU**.⁵

GTK	
www.gtk.org	
Tipo de programa	toolkit Paquete GNU declarative programming language lenguaje de programación widget
Desarrollador	Fundación GNOME Proyecto GNU
Lanzamiento	14 de abril de 1998
Última versión estable	3.24.17 (info)
Última versión en pruebas	3 de abril de 2020 (6 días)
Género	hp android
Programado en	Biblioteca de desarrollo C
Sistema operativo	Multiplataforma
Plataforma	multiplataforma
Licencia	GNU LGPL
Estado actual	Activo
Idiomas	Multilingüe

Ejemplos de programas que fueron desarrollados usando GTK

Implementaciones disponibles [editar]

Existe una gran variedad de lenguajes de programación con los cuales se puede usar GTK,⁸ aunque no en todos está disponible en su última versión. Entre los más usados están los siguientes:

- C++ (gtkmm)
- C# (Gtk Sharp)
- Java (java-gnome)
- Python (PyGTK)
- Javascript
- Julia
- Vala
- Perl

Algunas aplicaciones que usan GTK para desarrollar sus interfaces de usuario incluyen:

- AbiWord - Procesador de textos.
- CinePaint (ex FilmGimp) - Editor de gráficos animados en HDR.
- Ekiga (ex GnomeMeeting) - Software telefónico VoIP H.323/SIP.
- Evolution - Cliente de correo electrónico.
- Firefox - Navegador web.
- GIMP - Editor de gráficos.
- Gnumeric - Programa de hoja de cálculo.
- Chromium - Navegador Web basado en WebKit y desarrollado en gran medida por Google.
- GRAMPS - Software de genealogía.
- Inkscape - Editor de gráficos vectoriales SVG.
- K-3D - Programa de modelado 3D libre.
- Marionnet - Un simulador de red interactivo.
- Midori - Navegador Web ligero, forma parte del proyecto XFCE.
- Nero Linux - Un programa para la edición de discos.
- Pidgin - Cliente de mensajería instantánea.
- VMware Player - Máquina virtual.
- Wireshark - Capturador y analizador de paquetes de redes computacionales.

Ejemplo introductorio a la programación con GTK:

```
int
main (int    argc,
      char **argv)
{
    GtkApplication *app;
    int status;

    app = gtk_application_new ("gtk.example", G_APPLICATION_FLAGS_NONE);
    g_signal_connect (app, "activate", G_CALLBACK (activate), NULL);
    status = g_application_run (G_APPLICATION (app), argc, argv);
    g_object_unref (app);

    return status;
}
```

```
#include <gtk/gtk.h>

/*retrollamada ante clicks del botón 1*/
static void
imprimir_hola (GtkWidget *widget,
                 gpointer   data)
{
    g_print ("Hola, Mundo!\n");
}

/*retrollamada ante clicks del botón 2*/
void abrir_gedit (GtkWidget *widget,
                  gpointer   data)
{
    system("gedit --new-window Desde_Hola_GTK.txt");
}

static void
activate (GtkApplication *app,
          gpointer       user_data)
{
    GtkWidget *window;
    GtkWidget *button1, *button2;
    GtkWidget *button_box;
```

Ejemplo introductorio a la programación con GTK (cont.):

```
/*Creamos la ventana principal*/
window = gtk_application_window_new (app);
gtk_window_set_title (GTK_WINDOW (window), "HCMA - Curso 2020");
gtk_window_set_default_size (GTK_WINDOW (window), 300, 300);

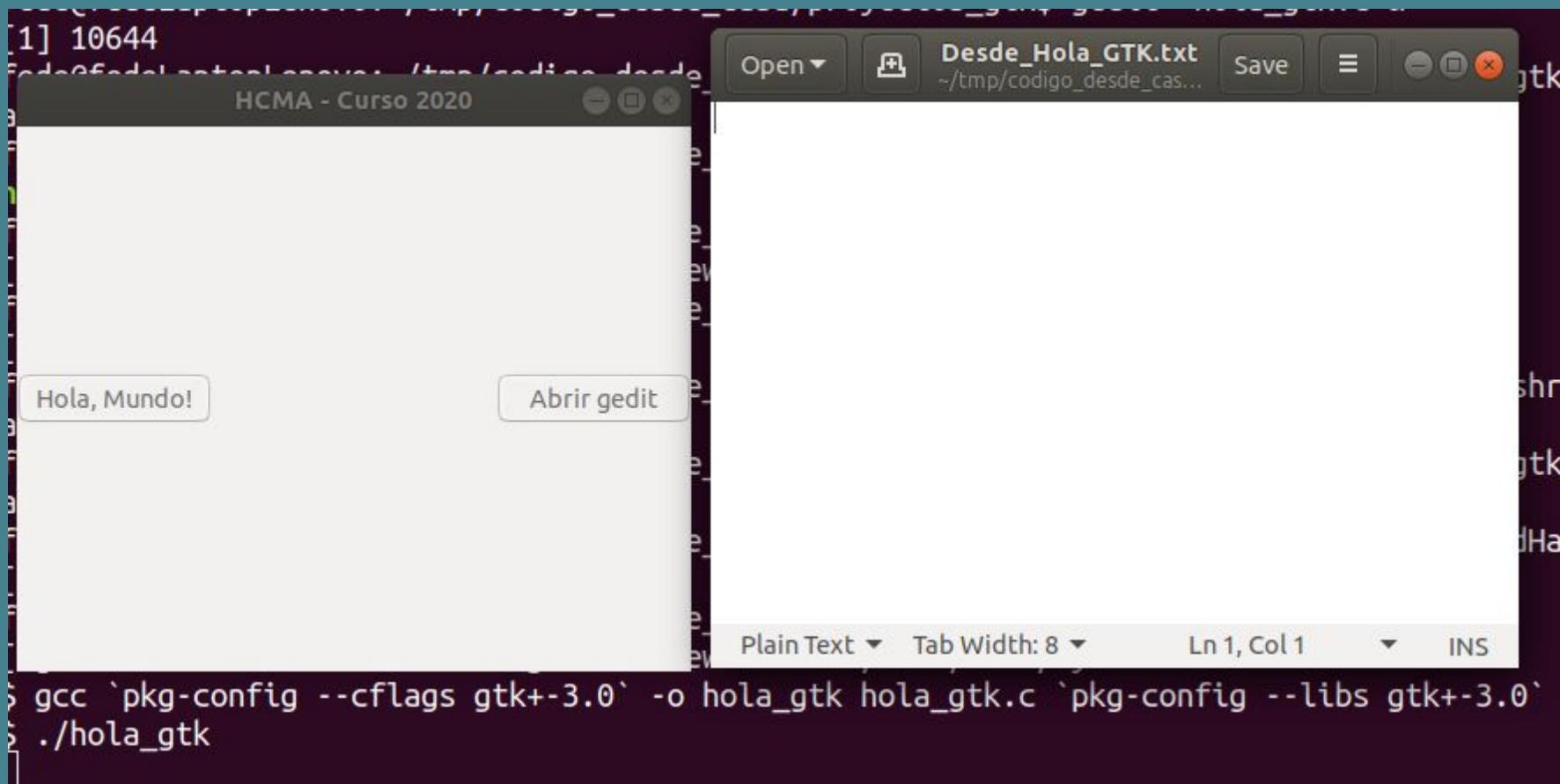
button_box = gtk_button_box_new (GTK_ORIENTATION_HORIZONTAL);
gtk_container_add (GTK_CONTAINER (window), button_box);

/*Agregamos un botón*/
button1 = gtk_button_new_with_label ("Hola, Mundo!");
g_signal_connect (button1, "clicked", G_CALLBACK (imprimir_hola), NULL);
g_signal_connect_swapped (button1, "clicked", G_CALLBACK (gtk_widget_destroy), window);
gtk_container_add (GTK_CONTAINER (button_box), button1);

/*Agregamos otro botón*/
button2 = gtk_button_new_with_label ("Abrir gedit");
g_signal_connect (button2, "clicked", G_CALLBACK (abrir_gedit), NULL);
gtk_container_add (GTK_CONTAINER (button_box), button2);

gtk_widget_show_all (window);
}
```

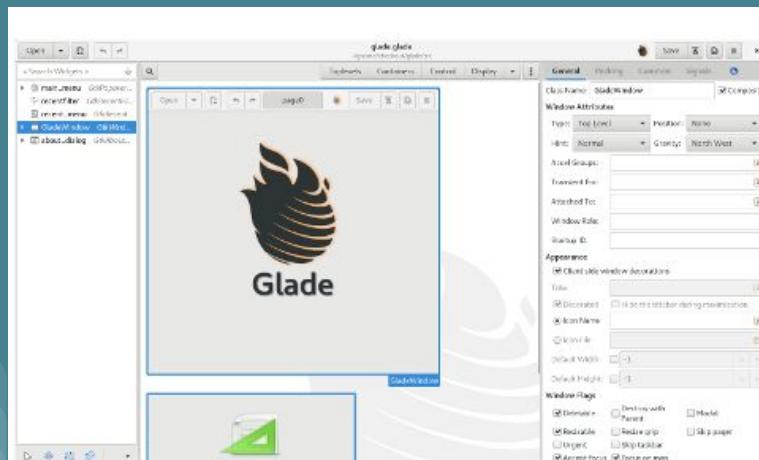
Ejecutando “hola_gtk”:



Programación Visual con GTK + Glade:

Glade (o Glade Interface Designer, que significa 'Diseñador de interfaces Glade') es una herramienta de desarrollo visual de interfaces gráficas mediante GTK/GNOME. Es independiente del lenguaje de programación y predeterminadamente no genera código fuente sino un archivo XML (ver sección GtkBuilder). La posibilidad de generar automáticamente código fuente fue descontinuada desde Glade versión 3.

Aunque tradicionalmente se ha utilizado de forma independiente, está totalmente integrado en Anjuta 2. Se encuentra bajo la licencia GPL. Para QT existe un proyecto similar, QtDesigner.



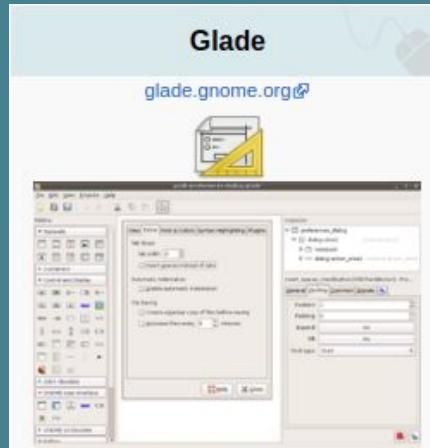
files can be used in numerous **programming languages** including C, C++, C#, Vala, Java, Perl, Python, and others.

What is Glade?

Glade is a RAD tool to enable quick & easy development of user interfaces for the **GTK+** toolkit and the **GNOME** desktop environment.

The user interfaces designed in Glade are saved as XML, and by using the **GtkBuilder** GTK+ object these can be loaded by applications dynamically as needed.

By using GtkBuilder, Glade XML



Glade

glade.gnome.org



Tipo de programa	desarrollo rápido de aplicaciones
Desarrollador	The GNOME Foundation
Lanzamiento	18 de abril de 1998
Última versión estable	3.22.0 ¹ (info)
Género	Diseñador de GUI
Programado en	C
Sistema operativo	Multiplataforma
Licencia	GPL
Estado actual	En desarrollo
Idiomas	Multilingüe
En español	✓ Si

Captura de Glade

desarrollo rápido de aplicaciones
software libre

The GNOME Foundation

18 de abril de 1998

3.22.0¹ ([info](#))

12 de marzo de 2018 (2 años y 28 días)

Diseñador de GUI

C

Multiplataforma

GPL

En desarrollo

Multilingüe

✓ Si

Apéndice:

Repasamos algunos conceptos que serán de utilidad para resolver la práctica 😊 🤝

Repaso: Método de Monte Carlo

El [método de Montecarlo](#)¹ es un [método no determinista](#) o estadístico numérico, usado para aproximar [expresiones matemáticas](#) complejas y costosas de evaluar con exactitud. El método se llamó así en referencia al [Casino de Montecarlo \(Mónaco\)](#) por ser “la capital del juego de azar”, al ser la [ruleta](#) un generador simple de [números aleatorios](#). El nombre y el desarrollo sistemático de los métodos de Montecarlo datan aproximadamente de [1944](#) y se mejoraron enormemente con el desarrollo de la computadora.

El método de Montecarlo proporciona soluciones aproximadas a una gran variedad de problemas matemáticos posibilitando la realización de experimentos con muestreos de números pseudoaleatorios en una computadora. El método es aplicable a cualquier tipo de problema, ya sea estocástico o determinista.

Cálculo de π por Montecarlo [\[editar\]](#)

Consideremos al círculo unitario [inscrito](#) en el cuadrado de lado 2 centrado en el origen. Dado que el cociente de sus áreas es $\frac{\pi}{4}$, el valor de π puede aproximarse usando Montecarlo de acuerdo al siguiente método:³ ²

1. Dibuja un círculo unitario, y al cuadrado de lado 2 que lo [inscribe](#).
2. Lanza un número n de [puntos aleatorios uniformes](#) dentro del cuadrado.
3. Cuenta el número de puntos dentro del círculo, i.e. puntos cuya distancia al origen es menor que 1.
4. El cociente de los puntos dentro del círculo dividido entre n es un estimado de, $\frac{\pi}{4}$. Multiplica el resultado por 4 para estimar π .

En este cálculo se tienen que hacer dos consideraciones importantes:

1. Si los puntos no están uniformemente distribuidos, el método es inválido.
2. La aproximación será pobre si solo se lanzan unos pocos puntos. En promedio, la aproximación mejora conforme se aumenta el número de puntos.

Repaso: integración numérica

Regla del trapecio compuesta [\[editar\]](#)

La **regla del trapecio compuesta** o **regla de los trapecios** es una forma de aproximar una integral definida utilizando n trapecios. En la formulación de este método se supone que f es continua y positiva en el intervalo $[a,b]$. De tal modo la integral definida $\int_a^b f(x) dx$ representa el área de la región delimitada por la gráfica de f y el eje x , desde $x=a$ hasta $x=b$. Primero se divide el intervalo $[a,b]$ en n subintervalos, cada uno de ancho $h = (b - a)/n$.

Después de realizar todo el proceso matemático se llega a la siguiente fórmula:

$$\int_a^b f(x) dx \sim \frac{h}{2} [f(a) + 2f(a + h) + 2f(a + 2h) + \dots + f(b)]$$

Donde $h = \frac{b - a}{n}$ y n es el número de divisiones.

La expresión anterior también se puede escribir como:

$$\int_a^b f(x) dx \sim \frac{b - a}{n} \left[\frac{f(a) + f(b)}{2} + \sum_{k=1}^{n-1} f\left(a + k \frac{b - a}{n}\right) \right]$$

El error en esta aproximación se corresponde con :

$$-\frac{(b - a)^3}{12n^2} f''(\xi)$$

Siendo n el número de subintervalos

Series de Fourier I

Serie de Fourier

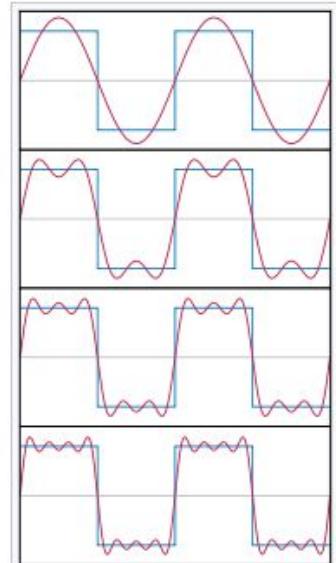
Una **serie de Fourier** es una serie infinita que converge puntualmente a una **función periódica y continua** a trozos (o por partes). Las series de Fourier constituyen la herramienta matemática básica del análisis de Fourier empleado para analizar funciones periódicas a través de la descomposición de dicha función en una suma infinita de funciones sinusoidales mucho más simples (como combinación de senos y cosenos con frecuencias enteras). El nombre se debe al matemático francés **Jean-Baptiste Joseph Fourier**, que desarrolló la teoría cuando estudiaba la **ecuación del calor**. Fue el primero que estudió tales series sistemáticamente, y publicó sus resultados iniciales en **1807** y **1811**. Esta área de investigación se llama algunas veces **análisis armónico**.

Es una aplicación usada en muchas ramas de la ingeniería, además de ser una herramienta sumamente útil en la teoría matemática abstracta. Sus áreas de aplicación incluyen análisis vibratorio, acústica, óptica, procesamiento de imágenes y señales, y compresión de datos. En ingeniería, para el caso de los sistemas de telecomunicaciones, y a través del uso de los componentes espectrales de frecuencia de una señal dada, se puede optimizar el diseño de un sistema para la señal portadora del mismo. Refiérase al uso de un analizador de espectros.

Las series de Fourier tienen la forma:

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos \frac{2n\pi}{T} t + b_n \sin \frac{2n\pi}{T} t \right]$$

donde a_n y b_n se denominan **coeficientes de Fourier** de la serie de Fourier de la función $f(t)$.



Las primeras cuatro aproximaciones para una función periódica escalonada

Series de Fourier II

Criterio de convergencia de la serie de Fourier

La función $f(x)$ es desarrollable en serie de Fourier en todo punto donde sea derivable y en todo punto de discontinuidad de primera especie en donde existan derivadas laterales. En este caso, la serie converge a

$$\frac{1}{2} [f(x_+) + f(x_-)]$$

La convergencia es uniforme en $[a_1, b_1]$ si además $f(x)$ es continua en (a, b) , con $a < a_1 < b_1 < b$.

Series de Fourier III

Cómo obtener los coeficientes de Fourier de una función periódica de periodo arbitrario?

1. Serie de Fourier en intervalo cualquiera. — Una serie de la forma

$$[99-1] \quad f(x) \sim \frac{1}{2}a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{nx}{\lambda} + b_n \sin \frac{nx}{\lambda} \right)$$

representa una función periódica con período $2\pi\lambda$. Los coeficientes, en lugar de [98-2], vendrán dados por

$$[99-2] \quad \begin{cases} a_n = \frac{1}{\pi\lambda} \int_{-\pi\lambda}^{\pi\lambda} f(t) \cos \frac{nt}{\lambda} dt , \\ b_n = \frac{1}{\pi\lambda} \int_{-\pi\lambda}^{\pi\lambda} f(t) \sin \frac{nt}{\lambda} dt , \quad (n = 0, 1, 2, \dots). \end{cases}$$

Referencias

- Como programar en C/C++ , H. M. Deitel and P. J. Deitel 2 ed. Pearson Prentice Hall.
- Numerical Recipes in C, The Art of Scientific Computing, W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Sec. Ed. Cambridge University Press.
- El lenguaje de programación, C , B.W. Kernighan and D. M. Ritchie, Prentice Hall Hispamérica
- www.cplusplus.com/reference/
- Unix Programación avanzada, F. M. Márquez 3ra ed. Alfaomega RA-MA Editorial.
- Procesos aleatorios, Y. Rozanov, MIR.
- Análisis Matemático, J. Rey Pastor, P. Pi Calleja y C. Trejo, Ed. Kapeluz.
- Desarrollo de aplicaciones Linux con GTK+ y GDK E. Harlow, Prentice Hall