

# Referencia Rápida de C

Sintaxis, punteros, gestión de memoria, esenciales de la biblioteca estándar

## Fundamentos

### Hola Mundo

```
#include <stdio.h>
int main(void) {
    printf("Hello, World!\n");
    return 0;
}
```

### Compilar y Ejecutar

```
gcc -o app main.c # compilar
gcc -Wall -Wextra -std=c17 main.c # estricto
./app # ejecutar
```

### Comentarios

```
// comentario de una línea (C99+)
/* comentario
multilínea */
```

## Tipos de Datos

### Tipos Primitivos

<b>char</b>	1 byte, carácter o entero pequeño
<b>short</b>	Al menos 16 bits
<b>int</b>	Al menos 16 bits (típicamente 32)
<b>long</b>	Al menos 32 bits
<b>long long</b>	Al menos 64 bits (C99+)
<b>float</b>	IEEE-754 de 32 bits
<b>double</b>	IEEE-754 de 64 bits
<b>_Bool / bool</b>	0 o 1 (usar <code>&lt;stdbool.h&gt;</code> para <code>bool</code> )

### Tipos de Ancho Fijo (stdint.h)

<b>int8_t, uint8_t</b>	Exactamente 8 bits con/sin signo
<b>int16_t, uint16_t</b>	Exactamente 16 bits
<b>int32_t, uint32_t</b>	Exactamente 32 bits
<b>int64_t, uint64_t</b>	Exactamente 64 bits
<b>size_t</b>	Sin signo, resultado de <code>sizeof</code>

### Conversión de Tipos

```
int i = (int)3.14; // conversión explícita
double d = (double)5 / 2; // 2.5, no 2
char c = (char)65; // 'A'
```

## Control de Flujo

### If / Else

```
if (x > 0) { printf("positive\n"); }
else if (x == 0) { printf("zero\n"); }
else { printf("negative\n"); }
```

### Switch

```
switch (choice) {
    case 1: printf("one\n"); break;
    case 2: printf("two\n"); break;
    default: printf("other\n");
}
```

### Bucles

```
for (int i = 0; i < 10; i++) { }
while (condition) { }
do { } while (condition);
```

## Sentencias de Salto

<b>break</b>	Salir del bucle o switch más cercano
<b>continue</b>	Saltar a la siguiente iteración
<b>return</b>	Salir de la función con valor opcional
<b>goto label</b>	Saltar a etiqueta (usar con moderación)

## Funciones

### Declaración y Definición

```
int add(int a, int b); // prototipo
int add(int a, int b) {
    return a + b;
}
```

### Punteros a Funciones

```
int (*op)(int, int) = add;
int result = op(3, 4); // llama a add(3, 4)
typedef int (*MathFn)(int, int);
MathFn fn = add;
```

### Funciones Estáticas

```
// visible solo dentro de esta unidad de traducción
static int helper(int x) {
    return x * 2;
}
```

## Punteros

### Fundamentos de Punteros

```
int x = 42;
int *p = &x; // p apunta a x
printf("%d\n", *p); // desreferenciar: 42
*p = 100; // x ahora es 100
```

### Aritmética de Punteros

```
int arr[] = {10, 20, 30};
int *p = arr;
printf("%d\n", *(p + 1)); // 20
printf("%d\n", p[2]); // 30 (igual que *(p+2))
```

### Patrones Comunes de Punteros

<b>int *p = NULL</b>	Puntero nulo (siempre inicializar)
<b>void *</b>	Puntero genérico (debe castearse para usar)
<b>const int *p</b>	Puntero a constante (no puede modificar el valor)
<b>int *const p</b>	Puntero constante (no puede reasignarse)
<b>int **pp</b>	Puntero a puntero (doble indirección)

## Arreglos y Cadenas

### Arreglos

```
int nums[5] = {1, 2, 3, 4, 5};
int matrix[2][3] = {{1,2,3}, {4,5,6}};
int len = sizeof(nums) / sizeof(nums[0]);
```

### Funciones de Cadena (string.h)

<b>strlen(s)</b>	Longitud (excluyendo terminador nulo)
<b>strcpy(dst, src)</b>	Copiar cadena (inseguro, preferir <code>strncpy</code> )
<b>strncpy(dst, src, n)</b>	Copiar como máximo n caracteres
<b>strcat(dst, src)</b>	Concatenar cadenas
<b>strcmp(a, b)</b>	Comparar: 0 si igual, <0 o >0 en caso contrario
<b>strchr(s, c)</b>	Encontrar primera ocurrencia del carácter
<b>strstr(haystack, needle)</b>	Encontrar subcadena

## Literales de Cadena

```
char greeting[] = "hello"; // arreglo mutable
const char *msg = "world"; // puntero a literal
char buf[64];
snprintf(buf, sizeof(buf), "%s %s", greeting, msg);
```

## Structs

### Definición y Uso

```
struct Point { double x; double y; };
struct Point p = {1.0, 2.0};
printf("(%g, %g)\n", p.x, p.y);
```

### Typedef

```
typedef struct {
    char name[50];
    int age;
} Person;
Person p = {"Alice", 30};
```

### Punteros a Struct

```
void set_age(Person *p, int age) {
    p->age = age; // operador flecha
}
```

### Enums y Unions

```
enum Color { RED, GREEN, BLUE };
union Data { int i; float f; char c; };
// Los miembros de union comparten la misma memoria
```

## Gestión de Memoria

### Asignación Dinámica (stdlib.h)

```
int *arr = malloc(10 * sizeof(int));
if (arr == NULL) { /* manejar error */ }
arr = realloc(arr, 20 * sizeof(int));
free(arr);
arr = NULL; // evitar puntero colgante
```

### Funciones de Asignación

<b>malloc(size)</b>	Asignar memoria sin inicializar
<b>calloc(count, size)</b>	Asignar e inicializar a cero
<b>realloc(ptr, size)</b>	Redimensionar bloque asignado
<b>free(ptr)</b>	Liberar memoria asignada

### Errores Comunes

<b>Memory leak</b>	Olvidar llamar a <code>free()</code> para la memoria asignada
<b>Double free</b>	Llamar a <code>free()</code> dos veces sobre el mismo puntero
<b>Puntero colgante</b>	Usar puntero después de <code>free()</code> — asignar NULL
<b>Buffer overflow</b>	Escribir más allá de los límites asignados

## E/S de Archivos

### Leer Archivos

```
FILE *f = fopen("data.txt", "r");
if (!f) { perror("open"); return 1; }
char line[256];
while (fgets(line, sizeof(line), f)) printf("%s", line);
fclose(f);
```

### Escribir en Archivos

```
FILE *f = fopen("out.txt", "w");
fprintf(f, "value: %d\n", 42);
fputs("hello\n", f);
fclose(f);
```

# Referencia Rápida de C

## Modos de Archivo

"r"	Lectura (el archivo debe existir)
"w"	Escritura (trunca o crea)
"a"	Añadir (crea si es necesario)
"rb", "wb"	Lectura / escritura binaria
"r+"	Lectura y escritura (el archivo debe existir)

## Preprocesador

### Directivas

```
#include <stdio.h> // encabezado del sistema
#include "myheader.h" // encabezado local
#define PI 3.14159
#define MAX(a, b) ((a) > (b) ? (a) : (b))
```

### Compilación Condicional

```
#ifdef DEBUG
    printf("debug: x = %d\n", x);
#endif
#ifndef HEADER_H /* include guard */
#define HEADER_H /* ... */ #endif
```

### Macros Comunes

__FILE__	Nombre del archivo fuente actual
__LINE__	Número de línea actual
__func__	Nombre de la función actual (C99+)
__DATE__	Cadena de fecha de compilación
NULL	Constante de puntero nulo
sizeof(x)	Tamaño del tipo o variable en bytes