

# C Tham Khảo Nhanh

Cú pháp, con trỏ, quản lý bộ nhớ, thư viện chuẩn

## Cơ Bản

### Hello World

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

### Biên Dịch & Chạy

```
gcc -o app main.c # biên dịch
gcc -Wall -Wextra -std=c17 main.c # nghiêm ngặt
./app # chạy
```

### Comments

```
// comment một dòng (C99+)
/* comment
nhiều dòng */
```

## Kiểu Dữ Liệu

### Kiểu Nguyên Thủy

<b>char</b>	1 byte, ký tự hoặc số nguyên nhỏ
<b>short</b>	ít nhất 16 bit
<b>int</b>	ít nhất 16 bit (thường 32)
<b>long</b>	ít nhất 32 bit
<b>long long</b>	ít nhất 64 bit (C99+)
<b>float</b>	IEEE-754 32-bit
<b>double</b>	IEEE-754 64-bit
<b>_Bool / bool</b>	0 hoặc 1 (dùng <b>&lt;stdbool.h&gt;</b> cho <b>bool</b> )

### Kiểu Độ Rộng Cố Định (stdint.h)

<b>int8_t, uint8_t</b>	Chính xác 8-bit có dấu / không dấu
<b>int16_t, uint16_t</b>	Chính xác 16-bit
<b>int32_t, uint32_t</b>	Chính xác 32-bit
<b>int64_t, uint64_t</b>	Chính xác 64-bit
<b>size_t</b>	Không dấu, kết quả của <b>sizeof</b>

### Ép Kiểu

```
int i = (int)3.14; // ép kiểu tường minh
double d = (double)5 / 2; // 2.5, không phải 2
char c = (char)65; // 'A'
```

## Luồng Điều Khiển

### If / Else

```
if (x > 0) { printf("dương\n"); }
else if (x == 0) { printf("không\n"); }
else { printf("âm\n"); }
```

### Switch

```
switch (choice) {
    case 1: printf("một\n"); break;
    case 2: printf("hai\n"); break;
    default: printf("khác\n");
}
```

### Vòng Lặp

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

## Câu Lệnh Jump

<b>break</b>	Thoát vòng lặp hoặc switch trong cùng
<b>continue</b>	Bỏ qua, chuyển sang lần lặp tiếp theo
<b>return</b>	Thoát hàm với giá trị tùy chọn
<b>goto label</b>	Nhảy đến nhãn (dùng cẩn thận)

## Hàm

### Khai Báo & Định Nghĩa

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

### Con Trỏ Hàm

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

### Hàm Static

```
// chỉ thấy được trong translation unit này
static int helper(int x) {
    return x * 2;
}
```

## Con Trỏ

### Cơ Bản Con Trỏ

```
int x = 42;
int *p = &x; // p trỏ đến x
printf("%d\n", *p); // hủy tham chiếu: 42
*p = 100; // x giờ là 100
```

### Số Học Con Trỏ

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

### Các Pattern Con Trỏ Thường Gặp

<b>int *p = NULL</b>	Con trỏ null (luôn khởi tạo)
<b>void *</b>	Con trỏ chung (phải ép kiểu để dùng)
<b>const int *p</b>	Con trỏ đến hằng (không thể sửa giá trị)
<b>int *const p</b>	Con trỏ hằng (không thể gắn lại con trỏ)
<b>int **pp</b>	Con trỏ đến con trỏ (hai cấp)

## Mảng & Chuỗi

### Mảng

```
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]);
```

### Hàm Chuỗi (string.h)

<b>strlen(s)</b>	Độ dài (không tính null terminator)
<b>strcpy(dst, src)</b>	Sao chép chuỗi (không an toàn, dùng <b>strncpy</b> )
<b>strncpy(dst, src, n)</b>	Sao chép tối đa n ký tự
<b>strcat(dst, src)</b>	Nối chuỗi
<b>strcmp(a, b)</b>	So sánh: 0 nếu bằng, <0 hoặc >0 khác nhau
<b>strchr(s, c)</b>	Tìm lần xuất hiện đầu tiên của ký tự
<b>strstr(haystack, needle)</b>	Tìm chuỗi con

## String Literals

```
char greeting[] = "hello"; // mảng có thể sửa
const char *msg = "world"; // con trỏ đến literal
char buf[64];
sprintf(buf, sizeof(buf), "%s %s", greeting, msg);
```

## Struct

### Định Nghĩa & Sử Dụng

```
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};
```

### Con Trỏ Struct

```
void set_age(Person *p, int age) {
    p->age = age; // toán tử mũi tên
}
```

### Enum & Union

```
enum Color { RED, GREEN, BLUE };
union Data { int i; float f; char c; };
// các member union dùng chung bộ nhớ
```

## Quản Lý Bộ Nhớ

### Cấp Phát Động (stdlib.h)

```
int *arr = malloc(10 * sizeof(int));
if (arr == NULL) { /* xử lý lỗi */ }
arr = realloc(arr, 20 * sizeof(int));
free(arr);
arr = NULL; // tránh dangling pointer
```

### Hàm Cấp Phát

<b>malloc(size)</b>	Cấp phát bộ nhớ chưa khởi tạo
<b>calloc(count, size)</b>	Cấp phát và khởi tạo về 0
<b>realloc(ptr, size)</b>	Thay đổi kích thước block đã cấp phát
<b>free(ptr)</b>	Giải phóng bộ nhớ đã cấp phát

### Các Lỗi Thường Gặp

<b>Memory leak</b>	Quên gọi <b>free()</b> cho bộ nhớ đã cấp phát
<b>Double free</b>	Gọi <b>free()</b> hai lần trên cùng con trỏ
<b>Dangling pointer</b>	Dùng con trỏ sau <b>free()</b> — đặt về NULL
<b>Buffer overflow</b>	Ghi vượt quá ranh giới đã cấp phát

## File I/O

### Đọc File

```
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);
```

### Ghi File

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

# C Tham Khảo Nhanh

## Chế Độ File

"r"	Đọc (file phải tồn tại)
"w"	Ghi (xóa nội dung hoặc tạo mới)
"a"	Thêm vào cuối (tạo nếu chưa có)
"rb", "wb"	Đọc / ghi nhị phân
"r+"	Đọc và ghi (file phải tồn tại)

## Preprocessor

### Directives

```
#include <stdio.h> // header hệ thống
#include "myheader.h" // header cục bộ
#define PI 3.14159
#define MAX(a, b) ((a) > (b) ? (a) : (b))
```

### Biên Dịch Có Điều Kiện

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

### Macro Thường Dùng

__FILE__	Tên file nguồn hiện tại
__LINE__	Số dòng hiện tại
__func__	Tên hàm hiện tại (C99+)
__DATE__	Chuỗi ngày biên dịch
NULL	Hằng con trỏ null
sizeof(x)	Kích thước kiểu hoặc biến tính bằng byte