C THAM KHẢO NHANH

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

Cơ Bản

Hello World

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

Biên Dịch & Chạy

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

Comments

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

Kiểu Dữ Liệu

Kiểu Nguyên Thủy

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

Kiểu Độ Rộng Cố Định (stdint.h)

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

Ép Kiểu

```c
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

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

Switch

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

Vòng Lặp

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

Câu Lệnh Jump

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

Hàm

Khai Báo & Định Nghĩa

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

Con Trỏ Hàm

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

Hàm Static

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

Con Trỏ

Cơ Bản Con Trỏ

```c
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

```c
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

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

Mảng & Chuỗi

Mảng

```c
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)

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

String Literals

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

Struct

Định Nghĩa & Sử Dụng

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

Typedef

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

Con Trỏ Struct

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

Enum & Union

```c
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)

```c
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

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

Các Lỗi Thường Gặp

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

File I/O

Đọc File

```c
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

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

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

```c
#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

```c
#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