C 快速参考

语法、指针、内存管理、标准库要点

基础

Hello World

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

编译与运行

```c
gcc -o app main.c          # compile
gcc -Wall -Wextra -std=c17 main.c  # strict
./app                       # run
```

注释

```c
// single-line comment (C99+)
/* multi-line
   comment */
```

数据类型

基本类型

char	1 字节，字符或小整数
short	至少 16 位
int	至少 16 位（通常 32 位）
long	至少 32 位
long long	至少 64 位（C99+）
float	32 位 IEEE-754
double	64 位 IEEE-754
_Bool / bool	**0** 或 **1**（用 **<stdbool.h>** 获得 **bool**）

固定宽度类型（stdint.h）

int8_t, uint8_t	精确 8 位有符号 / 无符号
int16_t, uint16_t	精确 16 位
int32_t, uint32_t	精确 32 位
int64_t, uint64_t	精确 64 位
size_t	无符号，**sizeof** 的结果类型

类型转换

```c
int i = (int)3.14;          // explicit cast
double d = (double)5 / 2;   // 2.5, not 2
char c = (char)65;          // 'A'
```

控制流

If / Else

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

Switch

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

循环

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

跳转语句

break	退出最内层循环或 switch
continue	跳到下一次迭代
return	退出函数并可选返回值
goto label	跳转到标签（尽量少用）

函数

声明与定义

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

函数指针

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

静态函数

```c
// visible only within this translation unit
static int helper(int x) {
    return x * 2;
}
```

指针

指针基础

```c
int x = 42;
int *p = &x;        // p points to x
printf("%d\n", *p); // dereference: 42
*p = 100;           // x is now 100
```

指针算术

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

常用指针模式

int *p = NULL	空指针（始终初始化）
void *	通用指针（使用时需转型）
const int *p	指向常量的指针（不能修改值）
int *const p	常量指针（不能重新赋值指针）
int **pp	指向指针的指针（双重间接）

数组与字符串

数组

```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]);
```

字符串函数（string.h）

strlen(s)	长度（不含终止符）
strcpy(dst, src)	复制字符串（不安全，推荐 **strncpy**）
strncpy(dst, src, n)	最多复制 n 个字符
strcat(dst, src)	连接字符串
strcmp(a, b)	比较：相等返回 0，否则 <0 或 >0
strchr(s, c)	查找字符第一次出现
strstr(haystack, needle)	查找子串

字符串字面量

```c
char greeting[] = "hello";     // mutable array
const char *msg = "world";     // pointer to literal
char buf[64];
snprintf(buf, sizeof(buf), "%s %s", greeting, msg);
```

结构体

定义与使用

```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};
```

结构体指针

```c
void set_age(Person *p, int age) {
    p->age = age;    // arrow operator
}
```

枚举与联合

```c
enum Color { RED, GREEN, BLUE };
union Data { int i; float f; char c; };
// union members share the same memory
```

内存管理

动态分配（stdlib.h）

```c
int *arr = malloc(10 * sizeof(int));
if (arr == NULL) { /* handle error */ }
arr = realloc(arr, 20 * sizeof(int));
free(arr);
arr = NULL;   // avoid dangling pointer
```

分配函数

malloc(size)	分配未初始化内存
calloc(count, size)	分配并零初始化
realloc(ptr, size)	调整已分配内存大小
free(ptr)	释放已分配内存

常见陷阱

内存泄漏	忘记 **free()** 已分配的内存
重复释放	对同一指针调用两次 **free()**
悬空指针	**free()** 后继续使用——释放后设为 NULL
缓冲区溢出	写入超出已分配边界

文件 I/O

读取文件

```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);
```

写入文件

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

C 快速参考

文件模式

`"r"`	只读（文件必须存在）
`"w"`	写入（截断或创建）
`"a"`	追加（不存在则创建）
`"rb", "wb"`	二进制读 / 写
`"r+"`	读写（文件必须存在）

预处理器

指令

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

条件编译

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

常用宏

`__FILE__`	当前源文件名
`__LINE__`	当前行号
`__func__`	当前函数名（C99+）
`__DATE__`	编译日期字符串
`NULL`	空指针常量
`sizeof(x)`	类型或变量的字节大小