C 빠른 참조

문법, 포인터, 메모리 관리, 표준 라이브러리 핵심

기본

Hello World

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

컴파일 & 실행

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

주석

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

데이터 타입

기본 타입

char	1바이트, 문자 또는 소형 정수
short	최소 16비트
int	최소 16비트 (일반적으로 32)
long	최소 32비트
long long	최소 64비트 (C99+)
float	32비트 IEEE-754
double	64비트 IEEE-754
_Bool / bool	**0** 또는 **1** (**<stdbool.h>**로 **bool** 사용)

고정 너비 타입 (stdint.h)

int8_t, uint8_t	정확히 8비트 부호 있음/없음
int16_t, uint16_t	정확히 16비트
int32_t, uint32_t	정확히 32비트
int64_t, uint64_t	정확히 64비트
size_t	부호 없음, **sizeof** 결과

타입 캐스팅

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

제어 흐름

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");
}
```

반복문

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

점프 구문

break	가장 안쪽 루프 또는 switch 종료
continue	다음 반복으로 건너뜀
return	선택적 값과 함께 함수 종료
goto label	레이블로 점프 (신중히 사용)

함수

선언 & 정의

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

함수 포인터

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

정적 함수

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

포인터

포인터 기본

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

포인터 산술

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

일반 포인터 패턴

int *p = NULL	null 포인터 (항상 초기화)
void *	범용 포인터 (사용 시 캐스팅 필요)
const int *p	상수 포인터 (값 수정 불가)
int *const p	상수 포인터 (포인터 재할당 불가)
int **pp	포인터의 포인터 (이중 간접)

배열 & 문자열

배열

```
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]);
```

문자열 함수 (string.h)

strlen(s)	길이 (null 종료자 제외)
strcpy(dst, src)	문자열 복사 (안전하지 않음, **strncpy** 권장)
strncpy(dst, src, n)	최대 n개 문자 복사
strcat(dst, src)	문자열 연결
strcmp(a, b)	비교: 같으면 0, 아니면 <0 또는 >0
strchr(s, c)	문자의 첫 번째 발생 위치 찾기
strstr(haystack, needle)	부분 문자열 찾기

문자열 리터럴

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

구조체

정의 & 사용

```
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};
```

구조체 포인터

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

열거형 & 공용체

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

메모리 관리

동적 할당 (stdlib.h)

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

할당 함수

malloc(size)	초기화되지 않은 메모리 할당
calloc(count, size)	할당 및 0으로 초기화
realloc(ptr, size)	이전에 할당된 블록 크기 조정
free(ptr)	할당된 메모리 해제

일반적인 함정

Memory leak	할당된 메모리를 **free()** 하지 않음
Double free	같은 포인터를 두 번 **free()** 호출
Dangling pointer	**free()** 후 포인터 사용 — NULL로 설정
Buffer overflow	할당된 경계를 넘어 쓰기

파일 I/O

파일 읽기

```
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);
```

파일 쓰기

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

C 빠른 참조

파일 모드

`"r"`	읽기 (파일 존재 필요)
`"w"`	쓰기 (잘라내거나 생성)
`"a"`	추가 (없으면 생성)
`"rb", "wb"`	바이너리 읽기 / 쓰기
`"r+"`	읽기 및 쓰기 (파일 존재 필요)

전처리기

지시어

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

조건부 컴파일

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

주요 매크로

`__FILE__`	현재 소스 파일명
`__LINE__`	현재 줄 번호
`__func__`	현재 함수 이름 (C99+)
`__DATE__`	컴파일 날짜 문자열
`NULL`	null 포인터 상수
`sizeof(x)`	타입 또는 변수의 바이트 크기