

In one line: Read a declaration **from the name outward: right first** ([], ()), **then left** (*, const). const guards what is **immediately left** of it (or right, if it is first). restrict promises “this pointer is the only way to that memory”. A pointer is valid only while its object lives.

Reading declarations

declaration	read as · allowed
const int *p	ptr to const int · p++ ok, *p=1 error
int const *p	identical to the line above
int *const p	const ptr to int · *p=1 ok, p++ error
const int *const p	neither moves nor writes
int **pp	ptr to ptr — a callee can reuse <i>your</i> ptr
int *a[5]	array of 5 pointers to int
int (*a)[5]	pointer to an array of 5 ints
int *f(void)	function returning int *
int (*f)(void)	pointer to function returning int
char *(*x[3])(int)	array of 3 ptrs to fn(int) → char *

Rule of thumb: const left of * = the data is read-only; right of * = the pointer is fixed. Registers: volatile uint32_t *const REG = (volatile uint32_t *)0x3FF4403C;

Function pointers + void *ctx

```
typedef void (*event_cb)(uint8_t evt, void *ctx);
typedef struct { event_cb cb; void *ctx; } listener;
static void on_evt(uint8_t evt, void *ctx) {
    counter *c = ctx;           // void* -> T*: no cast in C
    c->n++;                      // C's "closure": fn + state
}
static counter clicks;          // ctx must OUTLIVE the listener
listener l = { on_evt, &clicks };
l.cb(EVT_PRESS, l.ctx);         // == (*l.cb)(EVT_PRESS, l.ctx)
static const event_cb table[] = { [EVT_A] = on_a,
                                  [EVT_B] = on_b }; // dispatch
```

restrict — “I am the only door”

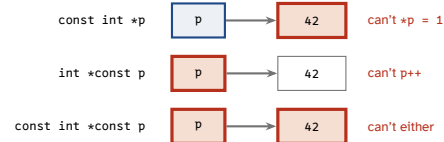
```
int f(int *a, int *b) { *a = 1; *b = 2; return *a; }
int g(int *restrict a, int *restrict b)
    { *a = 1; *b = 2; return *a; }
// f must reload *a (b may == a); g may just return 1
```

- Gain: values stay in registers, loads/stores reorder, loops **vectorise**. Breaking the promise = **UB**, no diagnostic.
- memcpy(void *restrict, const void *restrict, size_t) — overlap is UB. memmove has no restrict: copies in the safe direction. memmove(buf+2, buf, n) to shift in place.
- C99 keyword; C++ has only the __restrict extension.

Arrays, decay, arithmetic

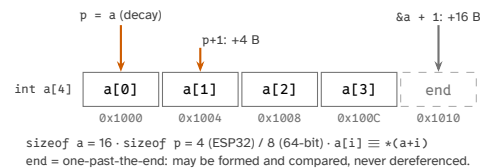
- An array name **decays** to &a[0] except under sizeof, unary &, and a string literal initialising a char array. &a is int (*)(4).
- Parameter int a[10] is int *a: inside, sizeof a = pointer size, so sizeof a / sizeof a[0] lies. Pass a length.
- p + n moves n * sizeof *p bytes; q - p = element count (ptrdiff_t), same array only. No arithmetic on void * in standard C (GCC treats it as bytes).

Picture — which part is read-only?



Red = read-only through this name. The 42 may still sit in RAM and change by another path.

Picture — decay, arithmetic, sizeof



Dangling pointers — and the fixes

- Return of a local:** char buf[32]; return buf; — the frame is popped; it may “work” until the next call overwrites it. Fix: **caller-provided buffer + size** (embedded default), static (not re-entrant), or malloc (caller frees).
- Use-after-free / double free:** free(p); p = NULL; (free(NULL) is a no-op).
- realloc may **move**: every old pointer dangles; p = realloc(p, n) leaks p on failure — use a temp.
- A ctx pointing at a stack variable, registered as a callback from a function that returns = dangling on the first event.

Interview traps

- ! Your Q3/Q18:** could not write “pointer to const” vs “const pointer” — say the name, go right, then left.
- ! Your Q6:** blamed the missing \0; char buf[32] = “hello world” has one. The bug is *lifetime*.
- ! Your Q15/Q31:** restrict = no-aliasing promise; overlap → memmove, never memcpy.
- ! const ≠ ROM:** const int *p may point at writable RAM.
- ! char ** → const char **** is *not* an implicit conversion.
- ! char *s = “hi”; s[0] = ‘H’;** is UB — use char s[] = “hi”;

Remember

Name, right, left. const guards its left neighbour. restrict = only door. A pointer never outlives its object.

Likely questions

- uint32_t *p; p+1? — address + 4.
- Return a string from a function, embedded? — caller passes buf, len.
- Why is memcpy faster than memmove? — may assume no overlap.
- When int **? — out-param that allocates, argv, jagged arrays.