C クイックリファレンス

構文、ポインタ、メモリ管理、標準ライブラリ必須事項

基礎

Hello World

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

コンパイル & 実行

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

コメント

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

データ型

プリミティブ型

char	1 バイト、文字または小整数
short	最低 16 ビット
int	最低 16 ビット（通常 32）
long	最低 32 ビット
long long	最低 64 ビット（C99+）
float	32 ビット IEEE-754
double	64 ビット IEEE-754
_Bool / bool	**0** または **1**（**bool** には **<stdbool.h>** を使用）

固定幅型（stdint.h）

int8_t, uint8_t	厳密な 8 ビット符号付き / なし
int16_t, uint16_t	厳密な 16 ビット
int32_t, uint32_t	厳密な 32 ビット
int64_t, uint64_t	厳密な 64 ビット
size_t	符号なし、**sizeof** の結果型

型キャスト

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

制御フロー

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");
}
```

ループ

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

ジャンプ文

break	最内側のループまたは switch を抜ける
continue	次のイテレーションへスキップ
return	オプションの値と共に関数を終了
goto label	ラベルへジャンプ（使用は最小限に）

関数

宣言 & 定義

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

関数ポインタ

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

static 関数

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

ポインタ

ポインタの基礎

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

ポインタ演算

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

ポインタのよくあるパターン

int *p = NULL	ヌルポインタ（常に初期化）
void *	汎用ポインタ（使用時にキャスト必要）
const int *p	定数へのポインタ（値の変更不可）
int *const p	定数ポインタ（ポインタ自体の再代入不可）
int **pp	ポインタへのポインタ（二重参照）

配列 & 文字列

配列

```
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]);
```

文字列関数（string.h）

strlen(s)	長さ（ヌル終端文字を除く）
strcpy(dst, src)	文字列をコピー（危険、**strncpy** を推奨）
strncpy(dst, src, n)	最大 n 文字をコピー
strcat(dst, src)	文字列を連結
strcmp(a, b)	比較: 等しければ 0、そうでなければ <0 か >0
strchr(s, c)	文字の最初の出現を検索
strstr(haystack, needle)	部分文字列を検索

文字列リテラル

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

構造体

定義 & 使用

```
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};
```

構造体ポインタ

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

enum & union

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

メモリ管理

動的確保（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
```

確保関数

malloc(size)	未初期化メモリを確保
calloc(count, size)	ゼロ初期化して確保
realloc(ptr, size)	確保済みブロックをリサイズ
free(ptr)	確保したメモリを解放

よくある落とし穴

メモリリーク	確保したメモリの **free()** 忘れ
二重解放	同じポインタを **free()** で 2 回解放
ダングリングポインタ	**free()** 後にポインタを使用 — NULL を代入
バッファオーバーフロー	確保した境界を超えて書き込み

ファイル I/O

ファイルの読み込み

```
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);
```

ファイルへの書き込み

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

C クイックリファレンス

ファイルモード

"r"	読み込み（ファイルが存在必要）
"w"	書き込み（切り捨てまたは作成）
"a"	追記（なければ作成）
"rb", "wb"	バイナリ読み込み / 書き込み
"r+"	読み書き（ファイルが存在必要）

プリプロセッサ

ディレクティブ

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

条件コンパイル

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

よく使うマクロ

__FILE__	現在のソースファイル名
__LINE__	現在の行番号
__func__	現在の関数名（C99+）
__DATE__	コンパイル日付の文字列
NULL	ヌルポインタ定数
sizeof(x)	型または変数のサイズ（バイト単位）