Referensi Cepat C

Sintaks, pointer, manajemen memori, esensial standard library

Dasar

Hello World

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

Kompilasi & Jalankan

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

Komentar

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

Tipe Data

Tipe Primitif

`char`	1 byte, karakter atau integer kecil
`short`	Minimal 16 bit
`int`	Minimal 16 bit (biasanya 32)
`long`	Minimal 32 bit
`long long`	Minimal 64 bit (C99+)
`float`	IEEE-754 32-bit
`double`	IEEE-754 64-bit
`_Bool / bool`	`0` atau `1` (gunakan `<stdbool.h>` untuk `bool`)

Tipe Lebar Tetap (stdint.h)

`int8_t, uint8_t`	Tepat 8-bit bertanda / tak bertanda
`int16_t, uint16_t`	Tepat 16-bit
`int32_t, uint32_t`	Tepat 32-bit
`int64_t, uint64_t`	Tepat 64-bit
`size_t`	Tak bertanda, hasil `sizeof`

Type Casting

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

Control Flow

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");
}
```

Loop

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

Pernyataan Lompat

`break`	Keluar dari loop atau switch terdalam
`continue`	Lanjut ke iterasi berikutnya
`return`	Keluar fungsi dengan nilai opsional
`goto label`	Lompat ke label (gunakan seperlunya)

Fungsi

Deklarasi & Definisi

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

Function Pointer

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

Fungsi Static

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

Pointer

Dasar Pointer

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

Aritmatika Pointer

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

Pola Pointer Umum

`int *p = NULL`	Pointer null (selalu inisialisasi)
`void *`	Pointer generik (harus di-cast untuk digunakan)
`const int *p`	Pointer ke konstanta (nilai tidak bisa diubah)
`int *const p`	Pointer konstanta (pointer tidak bisa di-reassign)
`int **pp`	Pointer ke pointer (double indirection)

Array & String

Array

```
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]);
```

Fungsi String (string.h)

`strlen(s)`	Panjang (tidak termasuk null terminator)
`strcpy(dst, src)`	Salin string (tidak aman, lebih baik `strncpy`)
`strncpy(dst, src, n)`	Salin maksimal n karakter
`strcat(dst, src)`	Gabungkan string
`strcmp(a, b)`	Bandingkan: 0 jika sama, <0 atau >0 jika tidak
`strchr(s, c)`	Temukan kemunculan pertama karakter
`strstr(haystack, needle)`	Temukan substring

String Literal

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

Struct

Definisi & Penggunaan

```
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};
```

Pointer ke Struct

```
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
```

Manajemen Memori

Alokasi Dinamis (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
```

Fungsi Alokasi

`malloc(size)`	Alokasi memori tanpa inisialisasi
`calloc(count, size)`	Alokasi dan inisialisasi nol
`realloc(ptr, size)`	Ubah ukuran blok yang sudah dialokasi
`free(ptr)`	Bebaskan memori yang dialokasi

Kesalahan Umum

`Memory leak`	Lupa memanggil `free()` pada memori yang dialokasi
`Double free`	Memanggil `free()` dua kali pada pointer yang sama
`Dangling pointer`	Menggunakan pointer setelah `free()` — set ke NULL
`Buffer overflow`	Menulis melampaui batas alokasi

File I/O

Membaca File

```
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);
```

Menulis ke File

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

Referensi Cepat C

Mode File

`"r"`	Baca (file harus ada)
`"w"`	Tulis (potong atau buat baru)
`"a"`	Tambahkan (buat jika tidak ada)
`"rb", "wb"`	Baca / tulis binary
`"r+"`	Baca dan tulis (file harus ada)

Preprocessor

Direktif

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

Kompilasi Kondisional

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

Macro Umum

`__FILE__`	Nama file sumber saat ini
`__LINE__`	Nomor baris saat ini
`__func__`	Nama fungsi saat ini (C99+)
`__DATE__`	String tanggal kompilasi
`NULL`	Konstanta pointer null
`sizeof(x)`	Ukuran tipe atau variabel dalam byte