C Language CheatSheet

This cheatsheet is designed to help you quickly revise **C syntax** before exams. Covers **basics, control flow, arrays, strings, pointers, functions, and file I/O**—the topics most commonly asked in practicals and viva.

---

1. Quick Start & Compilation

Basic Program

```c
#include <stdio.h>


int main(void) {
    printf("Hello, World!\n");
    return 0;
}
```

Compilation

```bash
# Compile and run
gcc program.c -o program
./program

# With warnings (recommended)
gcc -Wall program.c -o program
```

---

2. Data Types & Variables

Primitive Types

```
char c = 'A';           // 1 byte
int i = 42;             // Typically 4 bytes
float f = 3.14f;        // 4 bytes
double d = 3.14159;     // 8 bytes
```

Size and Ranges

```
printf("Size of int: %zu\n", sizeof(int));
```

---

3. Input / Output

Output with `printf`

```
printf("Integer: %d\n", 10);
printf("Float: %.2f\n", 3.14);
printf("Char: %c\n", 'A');
printf("String: %s\n", "Hello");
// This is a single line comment
/* This is a
   multi-line comment */
```

Input with `scanf`

```
int num;
scanf("%d", &num);

char name[50];
scanf("%49s", name);    // Avoid buffer overflow
```

gets and puts

```c
char str[100];
gets(str);
puts(str);
// NOTE: gets() is unsafe. Use fgets(str, sizeof(str), stdin) instead.
```

4. Control Flow

if-else

```c
if (a > b) {
    printf("a is greater");
} else {
    printf("b is greater");
}
```

switch

```c
switch (ch) {
    case 1: printf("One"); break;
    case 2: printf("Two"); break;
    default: printf("Other");
}
```

Loops

```c
// for loop
for (int i = 0; i < 5; i++) printf("%d ", i);

// while loop
int i = 0;
while (i < 5) i++;
```

```
// do-while loop
int j = 0;
do { j++; } while (j < 5);
```

5. Arrays

```
int arr[5] = {1, 2, 3, 4, 5};

// Array size
size_t size = sizeof(arr) / sizeof(arr[0]);

// Traversal
for (size_t i = 0; i < size; i++)
    printf("%d ", arr[i]);
```

6. Strings

```
#include <string.h>

char str1[20] = "Hello";
char str2[] = "World";

printf("Length: %zu\n", strlen(str1));
strcpy(str1, "Hi");          // Copy
strcat(str1, str2);          // Concatenate
if (strcmp(str1, str2) == 0) // Compare
    printf("Equal");
```

String Functions

Function	Description	Usage Example
`strlen`	Get string length	`size_t len = strlen(str);`
`strcpy`	Copy string	`strcpy(dest, src);`
`strncpy`	Copy n chars	`strncpy(dest, src, n);`
`strcat`	Concatenate strings	`strcat(dest, src);`
`strncat`	Concatenate n chars	`strncat(dest, src, n);`
`strcmp`	Compare strings	`strcmp(str1, str2);`
`strncmp`	Compare n chars	`strncmp(str1, str2, n);`
`strchr`	Find char in string	`strchr(str, 'a');`
`strrchr`	Find last char in string	`strrchr(str, 'a');`
`strstr`	Find substring	`strstr(str, "sub");`
`strtok`	Tokenize string	`strtok(str, " ,");`

7. Pointers

```
int x = 10;
int *ptr = &x;

printf("Value: %d\n", *ptr); // Dereference
*ptr = 20;                  // Modify value
```

---

8. Functions

```
// Declaration
int add(int a, int b);
```

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

// Call
int sum = add(5, 10);
```

Call by Reference (Swap Example)

```c
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

---

9. Structures

```c
struct Student {
    char name[50];
    int age;
};

struct Student s1 = {"John", 20};
printf("%s %d", s1.name, s1.age);
```

typedef

`typedef` is used to create an alias for a data type.

```c
typedef struct {
    char name[50];
    int age;
} Student;
```

```c
Student s1 = {"John", 20};
printf("%s %d", s1.name, s1.age);
```

---

10. File I/O

```c
#include <stdio.h>

FILE *fp = fopen("data.txt", "w");
fprintf(fp, "Hello File\n");
fclose(fp);

fp = fopen("data.txt", "r");
char line[100];
while (fgets(line, sizeof(line), fp))
    printf("%s", line);
fclose(fp);
```

---

11. Preprocessor Directives

```c
#include <stdio.h>
#define PI 3.14
#define MAX(a,b) ((a) > (b) ? (a) : (b))

#ifdef DEBUG
    printf("Debug info\n");
#endif
```

---

12. Command-Line Arguments

```c
int main(int argc, char *argv[]) {
    printf("Program: %s\n", argv[0]);
    for (int i = 1; i < argc; i++)
        printf("Arg %d: %s\n", i, argv[i]);
}
```

---

13. Memory Management

```c
#include <stdlib.h>

int *arr = (int *)malloc(5 * sizeof(int)); // Dynamic allocation using malloc
// int *arr = (int *)calloc(5, sizeof(int)); // Zero-initialized using calloc

if (arr == NULL) {
    // Handle allocation failure
}
// int *arr = realloc(arr, 10 * sizeof(int)); // Resize array using realloc
free(arr); // Deallocate memory
```

14. Quick Reference Tables

Format Specifiers

Type	Specifier
int	%d
unsigned int	%u
float/double	%f
char	%c
string	%s

Type	Specifier
hex	%x
octal	%o
pointer	%p

Escape Sequences

Sequence	Meaning
`\n`	New line
`\t`	Tab
`\\`	Backslash
`\"`	Double quote
`\'`	Single quote