C Programming Cheat Sheet

Pointers & Memory Management

A pointer is a variable that stores the address of another variable.

Pointer Declaration & Initialization

```c
int x = 10;
int *ptr = &x;  // Pointer storing the address of x
```

Pointer Dereferencing

```c
printf("%d", *ptr);  // Outputs the value of x (10)
```

Memory Allocation Functions

- `malloc()`: Allocates memory (uninitialized)
- `calloc()`: Allocates memory (initialized to 0)
- `realloc()`: Resizes allocated memory
- `free()`: Deallocates memory

Example:

```c
int *p = (int*) malloc(sizeof(int) * 5); // Allocates space for 5 integers
free(p);  // Releases allocated memory
```

Inbuilt Libraries (Based on C)

Standard Input/Output (stdio.h)

- `printf()` - Print output

- `scanf()` - Read input

String Handling (string.h)

- `strlen()` - Get string length

- `strcpy()` - Copy a string

- `strcmp()` - Compare strings

- `strcat()` - Concatenate strings

Math Operations (math.h)

- `sqrt()` - Square root

- `pow()` - Exponentiation

- `abs()` - Absolute value

Example:

```c
#include <math.h>

int y = pow(2, 3);  // y = 8
```

Call by Value vs Call by Reference

Call by Value

- The function gets a copy of the variable.

- Changes do not affect the original variable.

Example:

```c
void modify(int a) {
    a = a + 10;
}
int main() {
    int x = 5;
    modify(x);
    printf("%d", x);  // Output: 5
}
```

Call by Reference

- The function gets the address of the variable.
- Changes affect the original variable.

Example:

```c
void modify(int *a) {
    *a = *a + 10;
}
int main() {
    int x = 5;
    modify(&x);
    printf("%d", x);  // Output: 15
}
```

Recursion

A function calling itself, used for problems like factorial, Fibonacci series, etc.

Example (Factorial Calculation):

```c
int factorial(int n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
}
int main() {
    printf("%d", factorial(5));  // Output: 120
}
```

Iteration (Loops in C)

For Loop

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

While Loop

```c
int i = 0;
while(i < 5) {
    printf("%d ", i);
```

```
    i++;
}
```

Do-While Loop

```c
int i = 0;
do {
    printf("%d ", i);
    i++;
} while(i < 5);
```