C/C++ Cheatsheet

Basics

Printing

```
#include <stdio.h>  // C
#include <iostream> // C++

int main() {
    // C
    printf("Hello, World!\n");
    // C++
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Variables and Datatypes

```
int x = 5;          // Integer
float y = 3.14;     // Float
char name[] = "Alice"; // C-style string
bool is_valid = true; // Boolean (C++)
```

Control Flow

```
// If-Else
if (x > 0) {
    printf("Positive\n");
} else if (x == 0) {
    printf("Zero\n");
} else {
    printf("Negative\n");
}

// Loops
for (int i = 0; i < 5; i++) {
    printf("%d\n", i);
}

while (x > 0) {
    printf("%d\n", x);
    x--;
}
```

Functions

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

int main() {
    printf("%d\n", add(3, 4));  // Output: 7
    return 0;
}
```

Memory Management

```
// C version
int *arr = (int *)malloc(5 * sizeof(int));
arr[0] = 10;
free(arr);  // Free memory

// C++ version
int *arr = new int[5];
arr[0] = 10;
delete[] arr;  // Free memory
```

Data Structures

HashTable(C)

```
typedef struct {
    char *key;
    int val;
    struct Node *next;
} Node;

unsigned int hash(char *key) {
    unsigned int h = 0;
    while (*key) {
        h = h * 31 + *key++;
    }
    return h % TABLE_SIZE;
}

void insert(char *key, int val) {
    unsigned int idx = hash(key);
    Node *n = malloc(sizeof(Node));
    n->key = key; n->val = val;
    n->next = table[idx];
    table[idx] = n;
}

int search(char *key) {
    for (Node *n = table[hash(key)]; n; n = n->next)
        if (!strcmp(n->key, key)) {
            return n->val;
        )
    return -1;  // Not found
}
```

HashTable(C++)

```
#include <iostream>
#include <unordered_map>
using namespace std;

unordered_map<string, int> table;
table[key] = val;
int get = table[key];
```

Array

```
int nums[3] = {1, 2, 3};
int len = sizeof(nums) / sizeof(nums[0]);
printf("%d\n", nums[0]);  // Access first element
```

Pointer

```
int x = 10;
int *p = &x;
printf("%d\n", *p);  // Dereference pointer
```

Struct

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

struct Person p1 = {"Alice", 30};
printf("%s is %d years old.\n", p1.name, p1.age);
```

Typedef

```
typedef unsigned int uint;
uint x = 10;

typedef struct {
    char name[50];
    int age;
} Person;

Person p1 = {"Alice", 30};
printf("%s is %d years old.\n", p1.name, p1.age);
```

Algorithms

Efficient Sort (C)

```
int compare(const void *a, const void *b)
{
    return (*(int *)a - *(int *)b);
}

int *nums;
int len;
qsort(nums, len, sizeof(int), compare)
```

Efficient Sort (C++)

```
#include <algorithm>
#include <vector>

vector<int> nums;
sort(nums.begin(), nums.end())
// or stable_sort
```

Class (C++)

```
class Car {
public:
    string brand;
    string model;

    Car(string b, string m) {
        brand = b;
        model = m;
    }

    void drive() {
        cout << brand << " " << model
                << " is driving." << endl;
    }
};

int main() {
    Car myCar("Tesla", "Model S");
    myCar.drive();
    return 0;
}
```

Files

In C

```
FILE *file = fopen("file.txt", "w");
if (file) {
    fprintf(file, "Hello, File!\n");
    fclose(file);
}
```

In C++

```
#include <fstream>
using namespace std;

ofstream file("file.txt");
if (file.is_open()) {
    file << "Hello, File!\n";
    file.close();
}
```

STL (C++)

Vector

```
#include <vector>
using namespace std;

vector<int> nums = {1, 2, 3};
nums.push_back(4);
cout << nums[0] << endl;
```

Map

```
#include <map>
using namespace std;

map<string, int> ages;
ages["Alice"] = 30;
cout << ages["Alice"] << endl;
```

Set

```
#include <set>
using namespace std;

set<int> s = {3, 1, 4, 1, 2};
s.insert(5);
cout << *s.begin() << endl;
```

Queue

```
#include <queue>
using namespace std;

queue<int> q;
q.push(10); q.push(20);
cout << q.front() << endl;
q.pop();
```

Stack

```
#include <stack>
using namespace std;

stack<int> st;
st.push(10); st.push(20);
cout << st.top() << endl;
st.pop();
```

Lambda and Algorithms

```
#include <algorithm>
#include <vector>
using namespace std;

vector<int> nums = {1, 2, 3, 4, 5};
auto is_even = [](int x) { return x % 2 == 0; };
nums.erase(remove_if(nums.begin(),
    nums.end(), is_even), nums.end());
```

Input and Output

C

```
int x;
scanf("%d", &x);
printf("%d\n", x);
```

C++

```
int x;
cin >> x;
cout << x << endl;
```