C++ Cheat Sheet

1. Basic Syntax

- **Main Function**:

```
cpp
Copy code
#include <iostream>  // Include necessary header files

int main() {
    std::cout << "Hello, World!" << std::endl;  // Output to the console
    return 0;  // Return from main function
}
```

- **Comments**:
  - Single-line comment: `// This is a comment`
  - Multi-line comment:

```
cpp
Copy code
/* This is a
multi-line comment */
```

- **Variables and Data Types**:

```
cpp
Copy code
int x = 10;    // Integer
double y = 5.5; // Floating-point number
char z = 'A';   // Character
std::string name = "John"; // String
bool isTrue = true; // Boolean
```

2. Control Structures

- **If-Else**:

```
cpp
Copy code
if (x > 10) {
    std::cout << "x is greater than 10";
} else {
    std::cout << "x is less than or equal to 10";
}
```

- **Switch-Case**:

```
cpp
Copy code
switch (x) {
    case 1:
        std::cout << "x is 1";
        break;
    case 2:
        std::cout << "x is 2";
        break;
    default:
        std::cout << "x is neither 1 nor 2";
```

```
}
```

- **Loops**:
  - **For loop**:

```
cpp
Copy code
for (int i = 0; i < 5; i++) {
    std::cout << i << std::endl;
}
```

  - **While loop**:

```
cpp
Copy code
int i = 0;
while (i < 5) {
    std::cout << i << std::endl;
    i++;
}
```

  - **Do-While loop**:

```
cpp
Copy code
int i = 0;
do {
    std::cout << i << std::endl;
    i++;
} while (i < 5);
```

3. Functions

- **Function Declaration and Definition**:

```
cpp
Copy code
int add(int a, int b);  // Function declaration

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

- **Function Overloading**:

```
cpp
Copy code
int add(int a, int b) {
    return a + b;
}
double add(double a, double b) {
    return a + b;
}
```

- **Lambda Functions**:

```
cpp
Copy code
auto sum = [](int a, int b) { return a + b; };
std::cout << sum(5, 3);  // Outputs 8
```

4. Arrays

- **Declaration and Initialization**:

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

- **Accessing Elements**:

```
cpp
Copy code
std::cout << arr[0];  // Outputs 1
```

- **Multidimensional Arrays**:

```
cpp
Copy code
int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
std::cout << arr[1][2];  // Outputs 6
```

5. Pointers

- **Pointer Declaration and Initialization**:

```
cpp
Copy code
int x = 10;
int* ptr = &x;  // Pointer to integer
```

- **Dereferencing Pointers**:

```
cpp
Copy code
std::cout << *ptr;  // Outputs the value of x, 10
```

- **Pointer to Array**:

```
cpp
Copy code
int arr[3] = {1, 2, 3};
int* ptr = arr;
std::cout << *(ptr + 1);  // Outputs 2
```

6. Object-Oriented Programming (OOP)

- **Class Declaration and Definition**:

```
cpp
Copy code
class MyClass {
public:
    int x;
    MyClass(int val) { x = val; }  // Constructor
    void display() { std::cout << x << std::endl; }
};

int main() {
    MyClass obj(10);
    obj.display();  // Outputs 10
    return 0;
}
```

- **Inheritance**:

```
cpp
Copy code
class Animal {
public:
    void speak() { std::cout << "Animal speaks" << std::endl; }
};

class Dog : public Animal {
public:
    void speak() { std::cout << "Dog barks" << std::endl; }
};

int main() {
    Dog d;
    d.speak();  // Outputs "Dog barks"
    return 0;
}
```

- **Polymorphism**:

```
cpp
Copy code
class Base {
public:
    virtual void speak() { std::cout << "Base speaks" << std::endl; }
};

class Derived : public Base {
public:
    void speak() override { std::cout << "Derived speaks" << std::endl; }
};

int main() {
    Base* b = new Derived();
    b->speak();  // Outputs "Derived speaks"
    delete b;
    return 0;
}
```

- **Encapsulation**:

```
cpp
Copy code
class MyClass {
private:
    int x;
public:
    void setX(int val) { x = val; }
    int getX() { return x; }
};
```

- **Abstraction**:

```
cpp
Copy code
class AbstractClass {
public:
    virtual void show() = 0;  // Pure virtual function
};

class ConcreteClass : public AbstractClass {
```

```
public:
    void show() override { std::cout << "Concrete implementation" <<
std::endl; }
};
```

7. Standard Template Library (STL)

- **Vector**:

```
cpp
Copy code
#include <vector>
std::vector<int> vec = {1, 2, 3, 4};
vec.push_back(5);
std::cout << vec[2];  // Outputs 3
```

- **Map**:

```
cpp
Copy code
#include <map>
std::map<int, std::string> m;
m[1] = "One";
m[2] = "Two";
std::cout << m[1];  // Outputs "One"
```

- **Set**:

```
cpp
Copy code
#include <set>
std::set<int> s = {1, 2, 3};
s.insert(4);
std::cout << *s.begin();  // Outputs 1
```

- **Stack**:

```
cpp
Copy code
#include <stack>
std::stack<int> st;
st.push(10);
st.push(20);
std::cout << st.top();  // Outputs 20
```

- **Queue**:

```
cpp
Copy code
#include <queue>
std::queue<int> q;
q.push(10);
q.push(20);
std::cout << q.front();  // Outputs 10
```

8. File I/O

- **Reading from a file**:

```
cpp
Copy code
#include <fstream>
```

```
std::ifstream input("file.txt");
std::string line;
while (getline(input, line)) {
    std::cout << line << std::endl;
}
input.close();
```

- **Writing to a file**:

```
cpp
Copy code
#include <fstream>
std::ofstream output("file.txt");
output << "Hello, File!" << std::endl;
output.close();
```

---

9. Common C++ Functions

- **Finding the size of a container**:

```
cpp
Copy code
std::vector<int> v = {1, 2, 3};
std::cout << v.size();  // Outputs 3
```

- **Sort a vector**:

```
cpp
Copy code
#include <algorithm>
std::vector<int> v = {3, 1, 4, 1, 5};
std::sort(v.begin(), v.end());
```

---

10. Miscellaneous

- **Typecasting**:

```
cpp
Copy code
double pi = 3.14;
int intPi = (int)pi;  // Cast double to int
```

- **Exception Handling**:

```
cpp
Copy code
try {
    int x = 5 / 0;
} catch (const std::exception& e) {
    std::cout << "Exception: " << e.what() << std::endl;
}
```