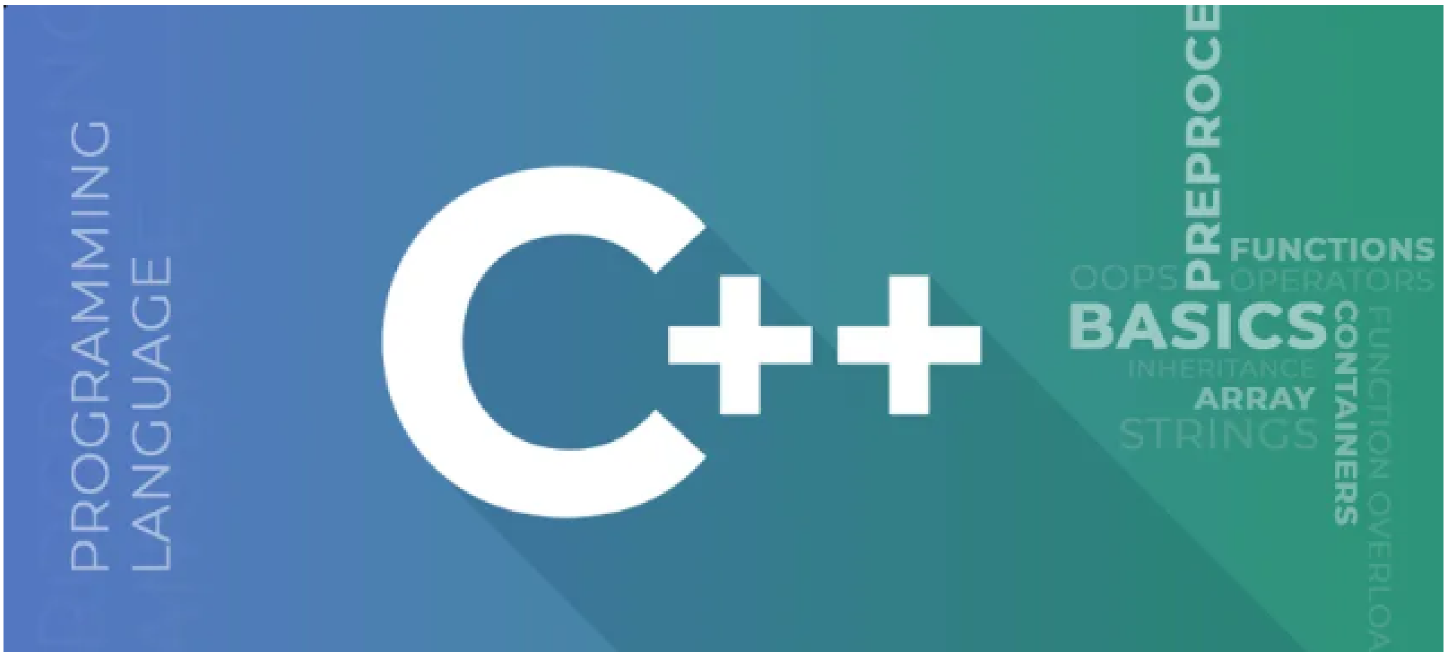

C++ Cheatsheet

1. Basic Syntax

- **Hello World:**

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

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

- **Comments:**

```
// Single line comment
/* Multi-line
   comment */
```

---

2. Data Types

- **Primitive Types:**

```
int a = 10;

float b = 10.5;

double c = 10.55;

char d = 'A';

bool e = true;
```

- **Type Modifiers:**

```
long int l = 100000;

unsigned int u = 50;
```

---

3. Variables and Constants

- **Variables:**

```
int x = 5;

int y = 10;
```

- **Constants:**

```
const int PI = 3.14;
```

---

4. Operators

- **Arithmetic Operators:**

```
+ - * / %
```

- **Relational Operators:**

```
== != > < >= <=
```

- **Logical Operators:**

```
&& || !
```

- **Assignment Operators:**

```
= += -= *= /= %=
```

- **Increment/Decrement:**

```
++ --
```

---

5. Control Structures

- **If-Else:**

```
if (condition) {
   // code
} else {
   // code
}
```

- **Switch:**

```
switch(variable) {
   case 1: // code
         break;
   case 2: // code
         break;
   default: // code
}
```

- **Loops:**
  - **For Loop:**

```
for (int i = 0; i < 10; i++) {
   // code
}
```

  - **While Loop:**

```
while (condition) {
   // code
}
```

- **Do-While Loop:**

```
do {
    // code
} while (condition);
```

---

6. Functions

- **Function Declaration & Definition:**

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


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

- **Default Arguments:**

```
int sum(int a, int b = 5);
```

- **Inline Functions:**

```
inline int square(int x) {
    return x * x;
}
```

---

7. Arrays and Strings

- **Arrays:**

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

- **Multi-dimensional Arrays:**

```
int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

- **Strings:**

```
#include <string>

string str = "Hello";
```

8. Pointers

- **Pointer Declaration:**

```
int *ptr;
```

- **Pointer Initialization:**

```
int var = 10;

int *ptr = &var;
```

- **Dereferencing:**

```
int value = *ptr;
```

- **Pointer to Pointer:**

```
int **ptr2 = &ptr;
```

9. References

- **Reference Declaration:**

```
int var = 10;

int &ref = var;
```

10. Dynamic Memory Allocation

- **Using new and delete:**

```
int *ptr = new int;

delete ptr;


int *arr = new int[10];

delete[] arr;
```

11. Structures

- **Defining and Using Structures:**

```
struct Person {
    string name;
    int age;
};

Person person1 = {"Alice", 30};
```

---

12. Classes and Objects

- **Defining a Class:**

```
class Car {
public:
    string brand;
    int year;

    void display() {
        cout << "Brand: " << brand << ", Year: " << year << endl;
    }
};

Car myCar = {"Toyota", 2010};
myCar.display();
```

- **Constructors and Destructors:**

```
class Car {
public:
    string brand;
    int year;
```

```
  Car(string b, int y) {  // Constructor
    brand = b;
    year = y;
  }

  ~Car() {  // Destructor
    cout << "Destructor called" << endl;
  }
};
```

- **Access Specifiers:**

```
class Example {
private:
  int privateVar;

protected:
  int protectedVar;

public:
  int publicVar;
};
```

---

13. Inheritance

- **Single Inheritance:**

```
class Base {
public:
    int baseVar;
};

class Derived : public Base {
public:
    int derivedVar;
};
```

- **Multiple Inheritance:**

```
class Parent1 {
public:
    int var1;
};

class Parent2 {
public:
    int var2;
};

class Child : public Parent1, public Parent2 {
};
```

- **Accessing Base Class Members:**

```
Derived d;
d.baseVar = 10;
d.derivedVar = 20;
```

---

14. Polymorphism

- **Function Overloading:**

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

double sum(double a, double b) {
    return a + b;
}
```

- **Operator Overloading:**

```
class Complex {
public:
    int real, imag;

    Complex operator + (const Complex &obj) {
        Complex temp;
        temp.real = real + obj.real;
        temp.imag = imag + obj.imag;
        return temp;
    }
};
```

- **Virtual Functions:**

```
class Base {
public:
    virtual void display() {
        cout << "Base display" << endl;
    }
};

class Derived : public Base {
public:
    void display() override {
        cout << "Derived display" << endl;
    }
};
```

---

15. Templates

- **Function Template:**

```
template <typename T>
T add(T a, T b) {
    return a + b;
}
```

- **Class Template:**

```
template <class T>
class Box {
public:
    T value;
    Box(T v) : value(v) {}
};
```

---

16. Exception Handling

- **Try-Catch Block:**

```
try {
   int num = 10 / 0;
} catch (exception &e) {
   cout << "Exception: " << e.what() << endl;
}
```

- **Throwing Exceptions:**

```
throw runtime_error("Error occurred");
```

---

17. File I/O

- **Reading from a File:**

```
#include <fstream>


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

- **Writing to a File:**

```
ofstream outfile("output.txt");
outfile << "Hello, File!" << endl;
outfile.close();
```

---

prepared by chhavi-rohilla
https://www.linkedin.com/in/chhavi-rohilla-607996251