📘 **C++ Programming Notes – AetherCode (Notion Template)**

🖥️ Cover Page

Course Title: C++ Programming – Object-Oriented Development & STL\ **Prepared By:** AetherCode Team\ **Website:** www.aethercode.com

> *"Code. Notes. Clarity."*

🕐 Table of Contents

Introduction

C++ is an object-oriented programming language developed by Bjarne Stroustrup as an extension of C. It supports OOP paradigms, making it suitable for large-scale software development while retaining the efficiency of C.

💡 Features of C++

- Object-Oriented Programming (OOP)
- Strongly typed and compiled
- Operator and function overloading
- Inheritance and polymorphism
- Standard Template Library (STL)
- Compatibility with C

🧰Structure of a C++ Program

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

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

- `#include <iostream>`: Header for I/O
- `using namespace std`: Avoids prefixing `std::`
- `cout`: Output stream

📊Data Types and Variables

Type	Example
int	int a = 10;
float	float f = 2.5;
double	double d = 2.718
char	char ch = 'A';
bool	bool flag = true;

🔒OOP Concepts

- **Encapsulation:** Binding data & methods
- **Abstraction:** Hide internal details
- **Inheritance:** Derive from base class
- **Polymorphism:** Many forms – overloading/overriding

🔧Classes & Objects

```cpp
class Student {
public:
    string name;
    int age;

    void display() {
        cout << name << " is " << age << " years old.";
```

```
    }
};
```

- Create object: `Student s1;`

🧱Constructors & Destructors

- Constructor initializes object.
- Destructor cleans up.

```
class Demo {
public:
    Demo() { cout << "Constructed"; }
    ~Demo() { cout << "Destructed"; }
};
```

🧬Inheritance

```
class Animal {
public:
    void speak() { cout << "Animal sound"; }
};

class Dog : public Animal {
public:
    void bark() { cout << "Woof"; }
};
```

- Types: Single, Multiple, Multilevel, Hierarchical, Hybrid

🔦Polymorphism

Compile-time:

- Function Overloading
- Operator Overloading

Runtime:

- Virtual functions
- Method Overriding using base class pointers

📦Encapsulation and Abstraction

```
class Account {
private:
    int balance;
public:
    void deposit(int amt) { balance += amt; }
    int getBalance() { return balance; }
};
```

Friend Functions & Static Members

```
class A {
    friend void show(A);
    static int count;
};
```

- `friend`: Allows access to private members
- `static`: Shared by all objects

Operator Overloading

```
class Complex {
public:
    int real, imag;
    Complex operator+(Complex c) {
        Complex temp;
        temp.real = real + c.real;
        temp.imag = imag + c.imag;
        return temp;
    }
};
```

🔦Function Overloading & Templates

```
int add(int a, int b);
float add(float a, float b);

// Templates
template <class T>
T maximum(T a, T b) {
```

```
    return (a > b) ? a : b;
}
```

📁File Handling

```
ofstream fout("file.txt");
fout << "AetherCode!";
fout.close();
```

- Streams: `ifstream`, `ofstream`, `fstream`

🔍Exception Handling

```
try {
    throw 404;
} catch (int e) {
    cout << "Error: " << e;
}
```

- Use `try-catch-finally`

📚STL – Standard Template Library

Component	Description
Vector	Dynamic array
List	Doubly linked list
Map	Key-value dictionary
Set	Unique collection
Stack	LIFO structure
Queue	FIFO structure

```
#include <vector>
vector<int> v = {1, 2, 3};
v.push_back(4);
```

🔗Summary

C++ blends procedural and OOP techniques for scalable, efficient, and reusable code. It is ideal for competitive programming, game development, simulations, and large system software.

Next: Dive into Java to explore platform independence and strong OOP.