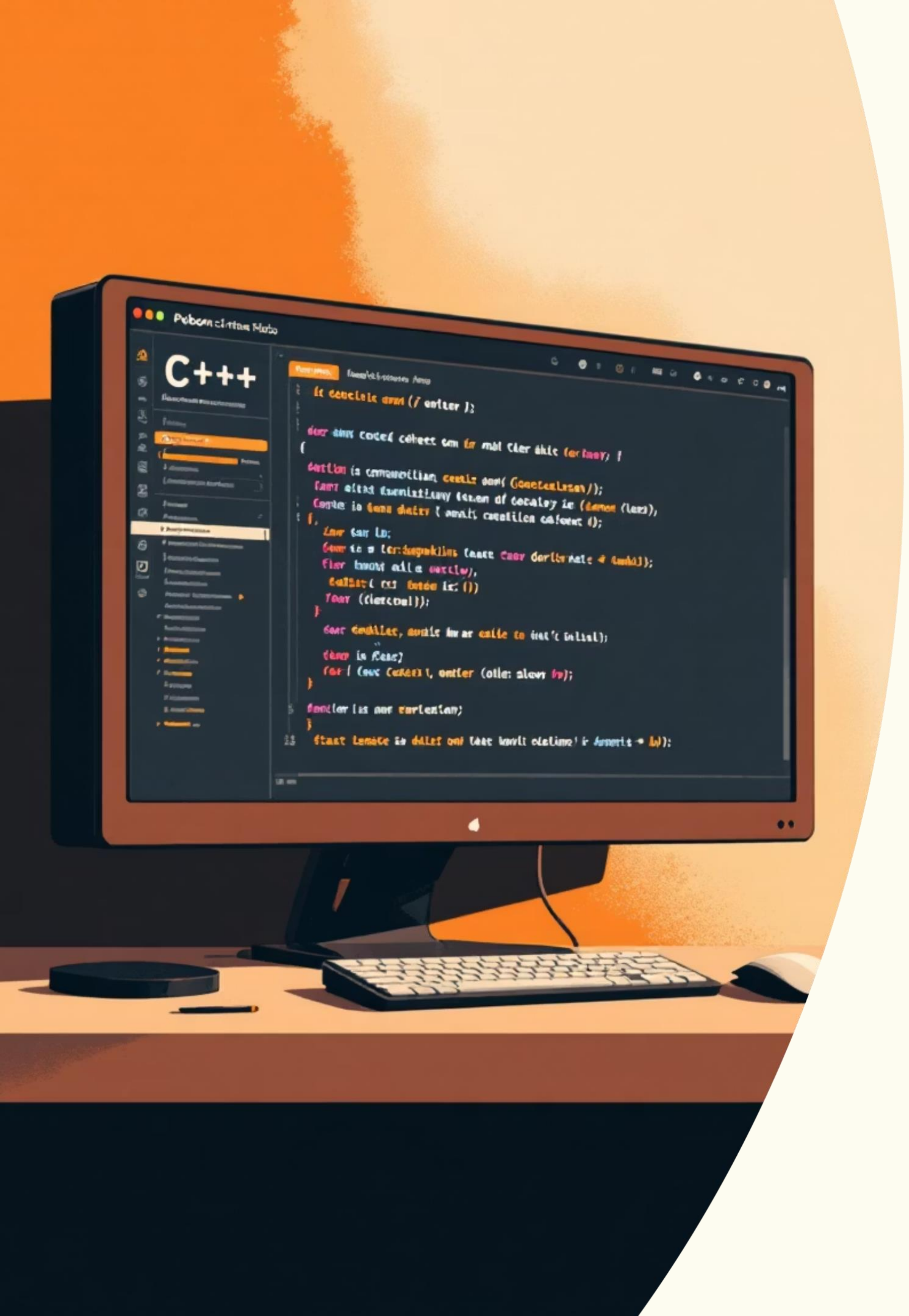
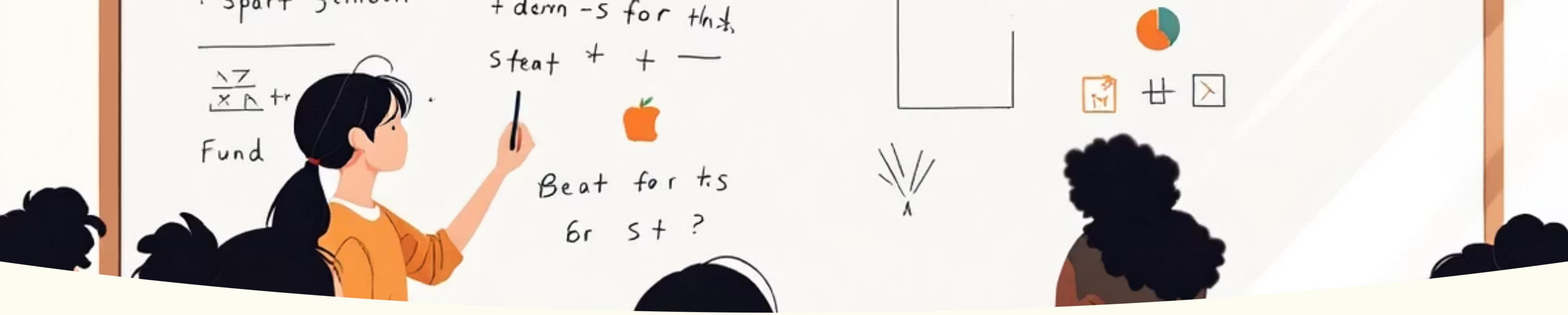


Static Keywords in C++

Understanding Static Variables & Functions



The Core Concept: Shared vs. Private

Static (Shared)

Think of a **classroom whiteboard** that everyone can see and use. There's only one board, and everyone shares the same information.

Static variables are shared across all instances - there's only one copy in memory.

Non-Static (Personal)

Think of a **personal notebook** where each student has their own copy. Each person's notes are private and independent.

Non-static variables work the same way - each function call or object gets its own separate copy.

Static data member

Static Data Members

A static data member is a variable declared with the `static` keyword. Class variables that are **shared by all objects** of that class. Only one copy exists, no matter how many objects you create.

Main Reasons for Using Static Data Members

- **Memory Efficiency**: Regular data members create a copy per object. Static data members exist only once for the entire class, regardless of object count. For example, tracking total objects created uses one counter instead of many.
- **Shared Information**: Store data common to all instances, like a global ID generator or configuration settings that all objects reference identically.
- **Class-Level Tracking**: Count object creation/destruction or maintain statistics without object-specific storage.

Static Data Members

Shared Across All Objects

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


class Student {
public:
   static int totalStudents;
   char name[20];


   Student(const char* n) {
      strcpy(name, n);
      totalStudents++;
   }
};


// Initialize static member
int Student::totalStudents = 0;


int main() {
   Student s1("Alice");
   Student s2("Bob");
   Student s3("Carol");


   cout << "Total Students: " << Student::totalStudents << endl;


   return 0;
}

//output:
Total Students: 3
```

All objects share the **same totalStudents variable**. Access it via the class name: Student::totalStudents. The code now uses C-style strings and constructor assignment, compatible with Turbo C++.

static member function

Static Member Functions

Functions that belong to the **class itself**, not to individual objects. They can only access static data members.

Key Reasons for using static member function:

- **No Object Required**: Call directly with `ClassName::function()` for utility tasks like math operations or validation.
- **Access Static Data**: Only they can directly modify static data members (like shared counters) without object instances.
- **Design Patterns**: Essential for Singleton (getInstance()), Factory methods, or class-wide helpers.
- **Memory Efficiency**: Single function in memory, not duplicated per object.

Static Member Functions

Belong to the Class, Not Objects

```
#include <iostream>
class Counter {
private:
    static int count;
public:
    static void increment() {
        count++;
    }
    static int getCount() {
        return count;
    }
};
// Initialize static variable
int Counter::count = 0;
int main() {
    Counter::increment();
    Counter::increment();

    cout << "Count: " << Counter::getCount();  // Output: 2
    return 0;
}
```

Rules

- Called using class name
- No **'this'** pointer
- Can only access static members
- Cannot be const or virtual

Made with GAMMA