**static variables**

Static variables have the property of preserving their value even after they are out of their scope. Static variables in C retain their value across function calls, offering persistence.

Static variables (like global variables) are initialized as 0 if not initialized explicitly

<https://youtube.com/shorts/Gq9jD7bw9LA?si=xXUASsREA3TO7PZf>

**static function**

A static function can be defined as one that has its scope limited to the file in which it is declared. This means that you cannot access the same from any other files in the program.

<https://www.youtube.com/shorts/rihJs_7Um0c>

A **static data member** is a variable that is shared across all instances of a class. It has only one copy for the entire class, and its memory is allocated at the class level. It can be used to track values like the number of objects created.

class Student {

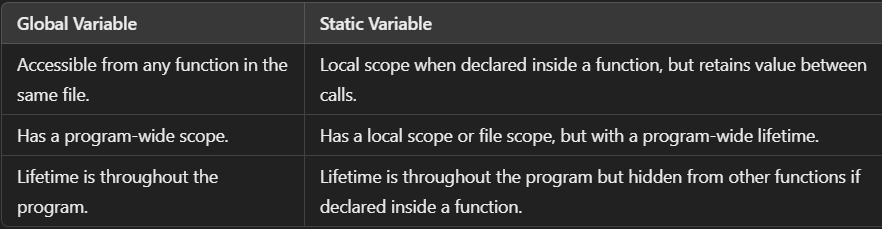
public:

static int count; // Static member

Student() { count++; }

};

int Student::count = 0;



Call by Value: Copies the actual parameter to the function, changes inside the function don't affect the original value.

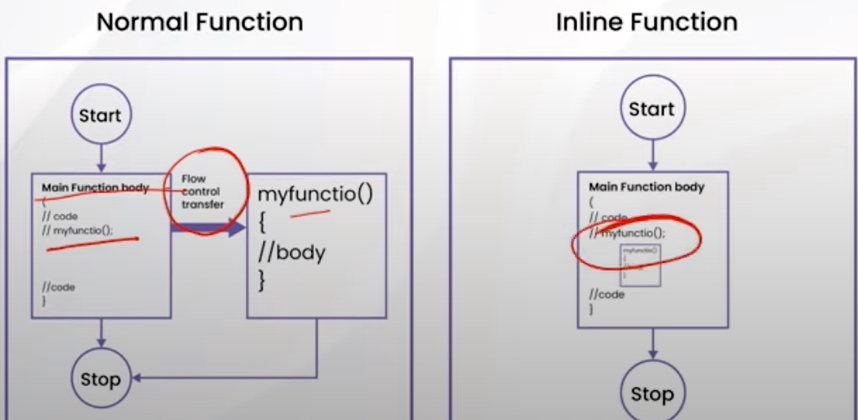
Call by Reference: Passes the memory address, allowing direct modification of the original value.

What is STL?

Standard Template Library provides built-in data structures like vectors, sets, and maps. It has three parts:

**Inline function**An **inline function** is a function where the compiler replaces the function call with the actual code, reducing the overhead of a normal function call. It is defined using the inline keyword.

Ex. Inline int cube(int x) { return x \* x \* x; }



**Dynamic Memory Allocation Functions:**

1. **malloc()**:  
   Allocates a block of memory of specified size. The memory is uninitialized.

c

Copy code

int \*ptr = (int \*)malloc(4 \* sizeof(int)); // Allocates memory for 4 integers

1. **calloc()**:  
   Allocates memory for an array of elements and initializes all bytes to zero.

c

Copy code

int \*ptr = (int \*)calloc(4, sizeof(int)); // Allocates and initializes memory for 4 integers

1. **realloc()**:  
   Resizes previously allocated memory. Can increase or decrease the size.

c

Copy code

ptr = (int \*)realloc(ptr, 6 \* sizeof(int)); // Resizes memory to hold 6 integers

1. **free()**:  
   Deallocates memory that was previously allocated. Prevents memory leaks.

c

Copy code

free(ptr); // Frees the allocated memory

**What is Data Structure?**

A **data structure** is a specialized format for organizing, processing, storing, and retrieving data. It defines the relationship between the data and the operations that can be performed on it. Data structures are essential in computer science for managing large amounts of data efficiently and are used in various applications, including databases, algorithms, and software development. Ex. Stack, Queue, Linked list.

Global variables are variables that are declared outside of all functions, typically at the top of a program. They can be accessed and modified from any part of the program, including all functions. Since they are not bound to any particular function or block, they exist for the entire lifetime of the program.

• Scope: Global variables have global scope, meaning they can be accessed from any part of the code.

• Lifetime: They exist throughout the program’s execution.

Local variables are variables that are declared within a function or a block of code, such as inside a loop or a conditional statement. These variables are only accessible within the scope in which they are declared and are destroyed when the function or block is exited.

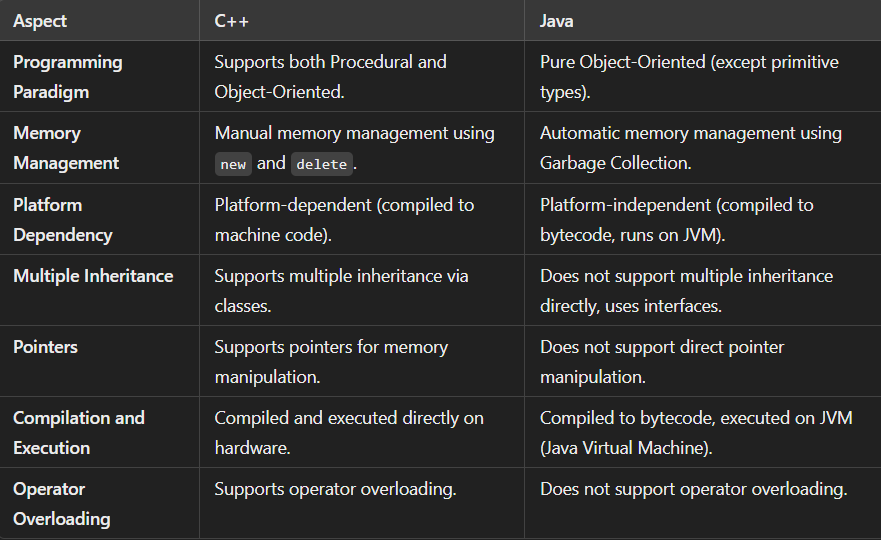
• Scope: Local variables have local scope, meaning they can only be accessed within the function or block where they are declared.

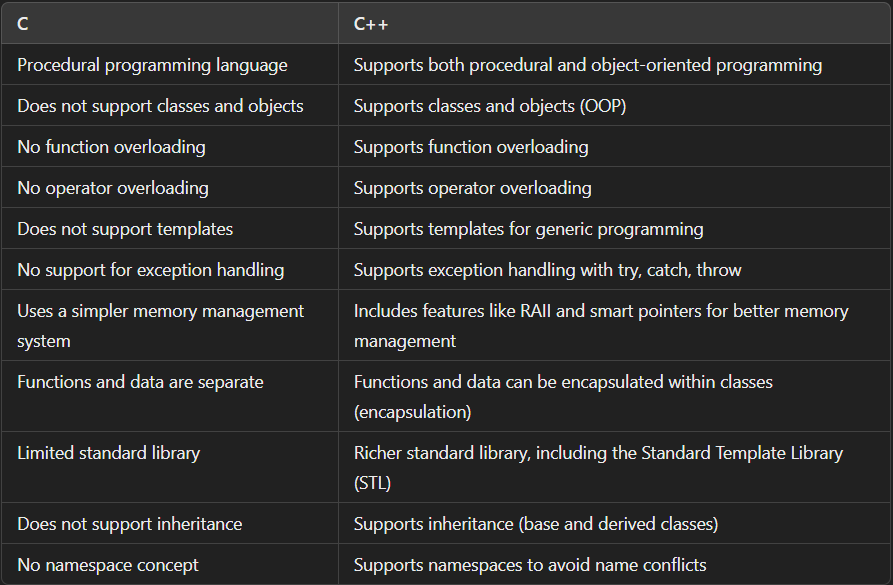
• Lifetime: They exist only while the function or block is executing.

**Quick Sort Overview**

**Exception handling** in C++ is a mechanism to handle runtime errors, allowing a program to deal with unexpected conditions (exceptions) gracefully, instead of crashing. C++ provides the try, catch, and throw keywords for exception handling.

* **try block**: This block contains the code that might generate an exception.
* **throw keyword**: Used to signal (throw) an exception.
* **catch block**: This block handles the exception thrown by the try block.





## C++ code execution

