# #23 DAY

## C++ COMPLETE BOOTCAMP

INSPIRE CLUB, MANIT BHOPAL

BRINGS

# C++ Complete Bootcamp

Learn How To Apply Problem Solving Skills

# Hello CPPBudddies

## Day No. 23

---

Welcome

To

# C++ COMPLETE BOOTCAMP

**Your Guide To A Solid Foundataion in C++**

Let us begin

# Struct vs Class

---

**C** has only structs

&

**C++** has both structs and classes

---

We can have functions inside C++ structs,

but not inside C structs

```
class Test {
    int x; // x is private
};
int main()
{
  Test t;
  t.x = 20; // compiler error because x is private
  getchar();
  return 0;
}
```

```
struct Test {
    int x; // x is public
};
int main()
{
  Test t;
  t.x = 20; // works fine because x is public
  getchar();
  return 0;
}
```

Members of a class are private by default and members of a structure are public by default.

# Class vs Struct

- Class can have null values but the structure can not have null values.
- Memory of structure is allocated in the stack while the memory of class is allocated in heap.
- Class requires constructor and destructor but the structure can not require it.
- Classes support polymorphism and also be inherited but the structure cannot be inherited.

```c
struct Employee
{
  char name[50];
  int age;
  float salary;
};

int main() {
  struct Employee e1 = {"John", 32, 4200};

  //accessing the values in the variable
  printf("Name: %s\n", e1.name);
  printf("Age : %d\n", e1.age);
  printf("Salary : %f\n", e1.salary);
}
```

# Static Variables

When a variable is declared as static, space for it gets allocated for the lifetime of the program.

Even if the function is called multiple times, space for the static variable is allocated only once and the value of variable in the previous call gets carried through the next function call.

This is useful for **implementing coroutines** in C/C++ or any other application where previous state of function needs to be stored.

```cpp
// C++ program to demonstrate
// the use of static Static
// variables in a Function
#include <iostream>
#include <string>
using namespace std;

void demo()
{
    // static variable
    static int count = 0;
    cout << count << " ";

    // value is updated and
    // will be carried to next
    // function calls
    count++;
}

int main()
{
    for (int i=0; i<5; i++)
        demo();
    return 0;
}
```

You can see in the above program that the variable count is declared as static.

So, its value is carried through the function calls.

The variable count is not getting initialized for every time the function is called.


Static Variables

# Static vs Normal Variables (in a class)

## Static variables in a class:

As the variables declared as static are initialized only once as they are allocated space in separate static storage so, the static variables in a class are shared by the objects.

There can not be multiple copies of same static variables for different objects.

There can not be multiple
copies of same static variables
for different objects.

Also because of this reason static
variables can not be initialized
using constructors.

All static data is initialized to zero when the first object is created, if no other initialization is present.

We **can't put it in the class definition** but it can be initialized outside the class as done in the following example by redeclaring the static variable, using the **scope resolution operator ::** to identify which class it belongs to.


# Static Data Members

# Static vs Normal Functions (in a class)

By declaring a function member as static, you make it independent of any particular object of the class.

# Static Function Members

A static member function can be called even if
no objects of the class exist
and the static functions are accessed using only
the class name
and the scope resolution operator ::

A static member function can only access

- static data member,
- other static member functions
- and any other functions

from outside the class.

Static member functions
have a class scope
and they do not have
access to the
this pointer of the class


# Static Member Functions

# Where can we define member functions of a class?

Member functions can be defined within the class definition

or

separately using scope resolution operator (::)

```
class Box {
   public:
      double length;      // Length of a box
      double breadth;     // Breadth of a box
      double height;      // Height of a box

      double getVolume(void) {
         return length * breadth * height;
      }
};
```

```
class Box {
   public:
      double length;         // Length of a box
      double breadth;        // Breadth of a box
      double height;         // Height of a box
      double getVolume(void);// Returns box volume
};

double Box::getVolume(void) {
   return length * breadth * height;
}
```

# THANK YOU

keep calm,
wear mask,
and
study hard

ISSUED IN PUBLIC INTEREST

whoami


**AKASH MAJI**
**[TCS DIGITAL]**
Your Mentor