DAY 00

Namespaces, classes, member functions, stdio streams, initialization lists, static, const, and some other basic stuff

Namespace :

- Namespace is a feature that allows you to group related declarations (e.g., variables, functions, and classes) together under a single identifier or name. This can help prevent naming conflicts and make your code more modular and easier to maintain.
- Using namespaces can help prevent naming conflicts between different parts of your program or between your program and libraries or other code that you may be using. It also makes it easier to organize and structure your code in a logical and modular way.

```
namespace one{
 int x;}
int main(){
using namespace one;  one::x                    // I want to access to this variable or function in this name space
.}
```

Avoiding naming conflicts : Suppose you have a variable named size in your program, and you also use a using directive for the std namespace. There could be a naming conflict with the std::size function, which is used to determine the size of an array. To avoid this, you can use the fully qualified name std::size to refer to the function .

Stdio stream :

- When data is written to a stream, it is first stored in a buffer before being written to the underlying device. This allows multiple write operations to be batched together into a single operation, which can improve performance by reducing the overhead associated with individual write operations. Similarly, when data is read from a stream, it is first read into a buffer before being processed by the program.
- C++ include an extensive standard library that provide IO, The iostream header file defines the classes and functions for input/output operations .
- we're using the cout object, which is defined in the std namespace within the iostream header file. We're using the scope resolution operator :: to access the cout object within the std namespacve.
- There are three main stream classes:

1. **std::cin** - the input stream class, which is used for reading input from the console or terminal window.
2. **std::cout** - the output stream class, which is used for writing output to the console or terminal window.
3. **std::cerr** - the error stream class, which is used for writing error messages to the console or terminal window.

- These stream classes are defined as objects of the istream, ostream, and ostream classes respectively, which are defined in the iostream header file.

OOP & Classes :

- OOP stands for Object-Oriented Programming, which is a programming paradigm that organizes code into objects that interact with each other. The main goal of OOP is to represent real-world entities and systems in a more natural and intuitive way, making it easier to understand, maintain, and modify the code.
- Class is code that describes a particular type of object . It specifies the data that an object can hold (the object's fields), and the action that an object can perform (the object's methods).
- You can think of a class as a code "blueprint" that can be used to create a particular type of object.

Why we need class : To add new data to all objects , no manually.

- A class is typically declared in a header file, and its member functions are implemented in a corresponding source file. The data members of a class can be either private, protected or public.
  - Private members can only be accessed by member functions of the same class.
  - Protected members can be accessed by member functions of the same class or by member functions of derived classes.
  - Public members can be accessed by any code that has access to an object of the class.

Data members private : By making data members private, you can ensure that the values they store can only be accessed and modified through the public member functions of the class. This allows you to control how the data is accessed and modified, and prevents external code from making invalid or inappropriate changes to the object's state.

Static :

Static Data Member :

- Static data member is a member of a class that is shared by all objects of that class. It is not associated with any particular instance of the class and is shared by all instances of the class.

```
class MyClass {
  public:
    static int count;      // static data member declaration
};
```

```
MyClass::count = 10;          // set the value of the static data member

int x = MyClass::count;       // access the value of the static data member

int MyClass::count = 0;       // static data member definition outside class
```

- Static data members in C++ **are stored in the static data segment of the memory**, which is a part of the program's memory allocation. The static data segment is separate from the stack and heap segments and is allocated when the program is loaded into memory.
- Since static data members are not associated with any instance of the class, they do not need to be stored within any particular object's memory. Instead, a single copy of the static data member is created and shared among all instances of the class. This means that the static data member has a fixed memory location throughout the program's execution, and all instances of the class access that same memory location when accessing the static data member.
- When a program terminates, the static data segment is deallocated along with all other program memory, and the static data member is destroyed.

Advantage of using private static data members is that they are shared by all instances of the class, which can help to conserve memory by eliminating redundant data storage. For example, if multiple objects of a class need to access the same data, it is more efficient to store the data as a private static data member instead of duplicating the data in each object.

Static Member Function :

- Static member function is a member function that belongs to the class and not to the object of the class. It can be called using the class name without creating an object of the class.
- Note that static member functions can only access static data members and other static member functions of the class. They cannot access non-static data members or member functions of the class.

```
class MyClass {
  public:
    static void myStaticFunction();          // static member function declaration
};


MyClass::myStaticFunction();                 // call the static member function
```

Const Data Member :

- Const data member is a member of a class that is declared as const and whose value cannot be changed once it has been initialized.
- Const data member can be initialized in two ways:

1. Using the member initialization list in the constructor:

```
class MyClass {
public:
    MyClass(int val) : const_member(val) {
        // Other constructor code
    }
private:
    const int const_member;
};
```

2 . Using a default member initializer:

```
class MyClass {
public:
    MyClass() {
        // Other constructor code
    }
private:
    const int const_member = 42;
};
```

Pointer this:

- The this pointer in C++ is a pointer that points to the object on which a member function is being called. It is an implicit pointer that is automatically created by the compiler and is passed as a hidden argument to all non-static member functions of a class.
- The main purpose of the this pointer is to allow the member functions of a class to access the data members and other member functions of the object on which they are called. When a member function is called on an object, the this pointer points to that object, allowing the function to access its data members and other member functions.

```
class MyClass {
public:
    void setX(int x) {
        this->x = x;
    }
private:
    int x;
};
```

```
int main() {
    MyClass obj;
    obj.setX(10);
    return 0;
}
```

- Without the this pointer, the setX function would not know which object's x data member to set. By using the this pointer, the function is able to access the correct object's data member and set its value.