



# SCOPE IN C++

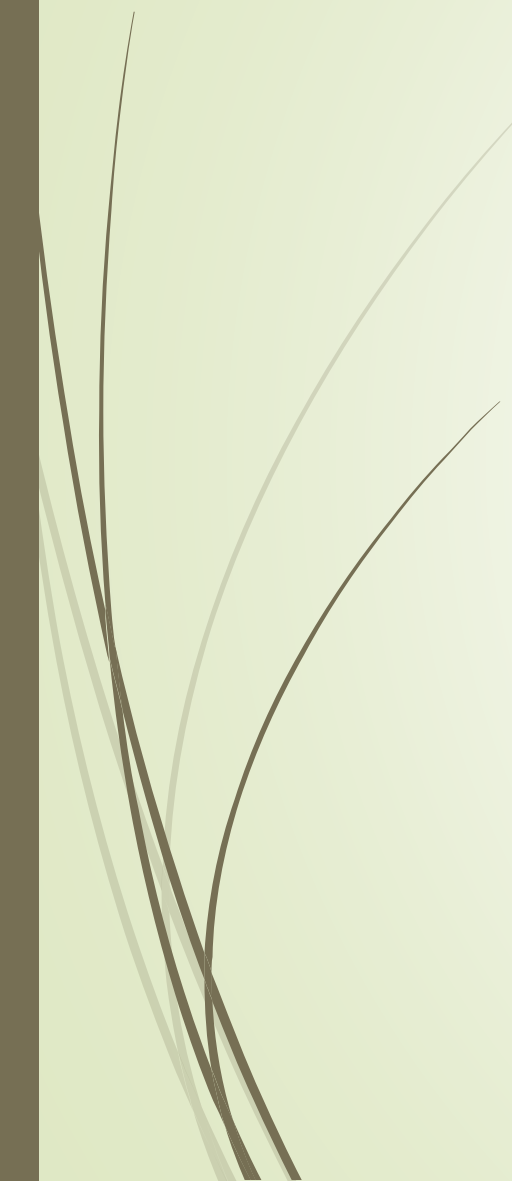


PRESENTED BY – ANJALI SONA  
DEPARTMENT OF COMPUTER SCIENCE

- 
- 
- ▶ When you declare a program element such as a class, function, or variable, its name can only be "seen" and used in certain parts of your program. The context in which a name is visible is called its scope. For example, if you declare a variable *x* within a function, *x* is only visible within that function body. It has *local scope*. You may have other variables by the same name in your program; as long as they are in different scopes, they do not violate the One Definition Rule and no error is raised.
  - ▶ For automatic non-static variables, scope also determines when they are created and destroyed in program memory.



# There are six kinds of scope:

1. **Global scope**
  2. **Namespace scope**
  3. **Local scope**
  4. **Class scope**
  5. **Statement scope**
  6. **Function scope**
- 



# 1. Global scope

A global name is one that is declared outside of any class, function, or namespace. However, in C++ even these names exist with an implicit global namespace. The scope of global names extends from the point of declaration to the end of the file in which they are declared. For global names, visibility is also governed by the rules of linkage which determine whether the name is visible in other files in the program.



## 2.Namespace scope

A name that is declared within a namespace, outside of any class or enum definition or function block, is visible from its point of declaration to the end of the namespace. A namespace may be defined in multiple blocks across different files.



## 3. Local scope

A name declared within a function or lambda, including the parameter names, have local scope. They are often referred to as "locals". They are only visible from their point of declaration to the end of the function or lambda body. Local scope is a kind of block scope, which is discussed later in this article.

## 4. Class scope

Names of class members have class scope, which extends throughout the class definition regardless of the point of declaration. Class member accessibility is further controlled by the `public`, `private`, and `protected` keywords.

Public or protected members can be accessed only by using the member-selection operators (`.` or `->`) or pointer-to-member operators (`.*` or `->*`).



## 5. Statement scope

Names declared in a `for`, `if`, `while`, or `switch` statement are visible until the end of the statement block.





## 6. Function scope



A label has function scope, which means it is visible throughout a function body even before its point of declaration.

Function scope makes it possible to write statements like `goto cleanup` before the `cleanup` label is declared.



# Abstraction in C++


- ▶ Data abstraction is one of the most essential and important feature of object oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.
- ▶ Consider a real life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is.

- 
- 
- **Abstraction using Classes:** We can implement Abstraction in C++ using classes. Class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to outside world and which is not.
  - **Abstraction in Header files:** One more type of abstraction in C++ can be header files. For example, consider the `pow()` method present in `math.h` header file. Whenever we need to calculate power of a number, we simply call the function `pow()` present in the `math.h` header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating power of numbers.



# Abstraction using access specifiers

- ▶ Access specifiers are the main pillar of implementing abstraction in C++. We can use access specifiers to enforce restrictions on class members. For example:
- ▶ Members declared as **public** in a class, can be accessed from anywhere in the program.
- ▶ Members declared as **private** in a class, can be accessed only from within the class. They are not allowed to be accessed from any part of code outside the class.
- ▶ We can easily implement abstraction using the above two features provided by access specifiers. Say, the members that defines the internal implementation can be marked as private in a class. And the important information needed to be given to the outside world can be marked as public. And these public members can access the private members as they are inside the class.




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

class implementAbstraction
{
    private:
        int a, b;

    public:

        // method to set values of
        // private members
        void set(int x, int y)
        {
            a = x;
            b = y;
        }

        void display()
        {
            cout<<"a = " <<a << endl;
            cout<<"b = " << b << endl;
        }
};
```



```
int main()
{
    implementAbstraction obj;
    obj.set(10, 20);
    obj.display();
    return 0;
}
```

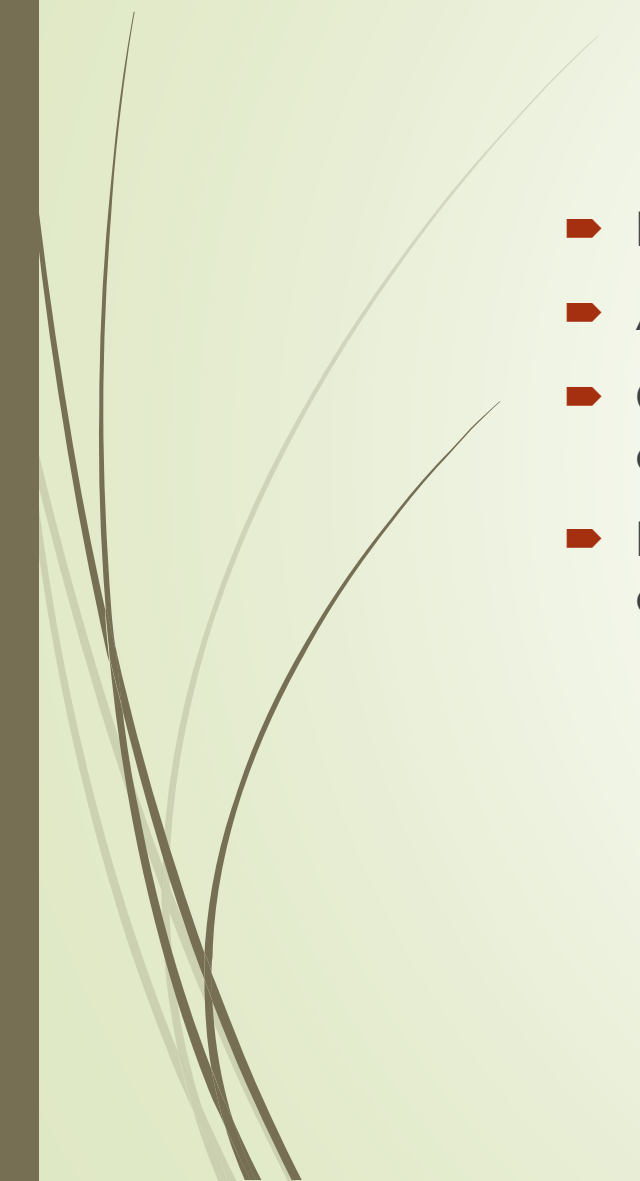
Output:

a = 10

b = 20



# Advantages of Data Abstraction:

- ▶ Helps the user to avoid writing the low level code
  - ▶ Avoids code duplication and increases reusability.
  - ▶ Can change internal implementation of class independently without affecting the user.
  - ▶ Helps to increase security of an application or program as only important details are provided to the user.
- 



THANK YOU