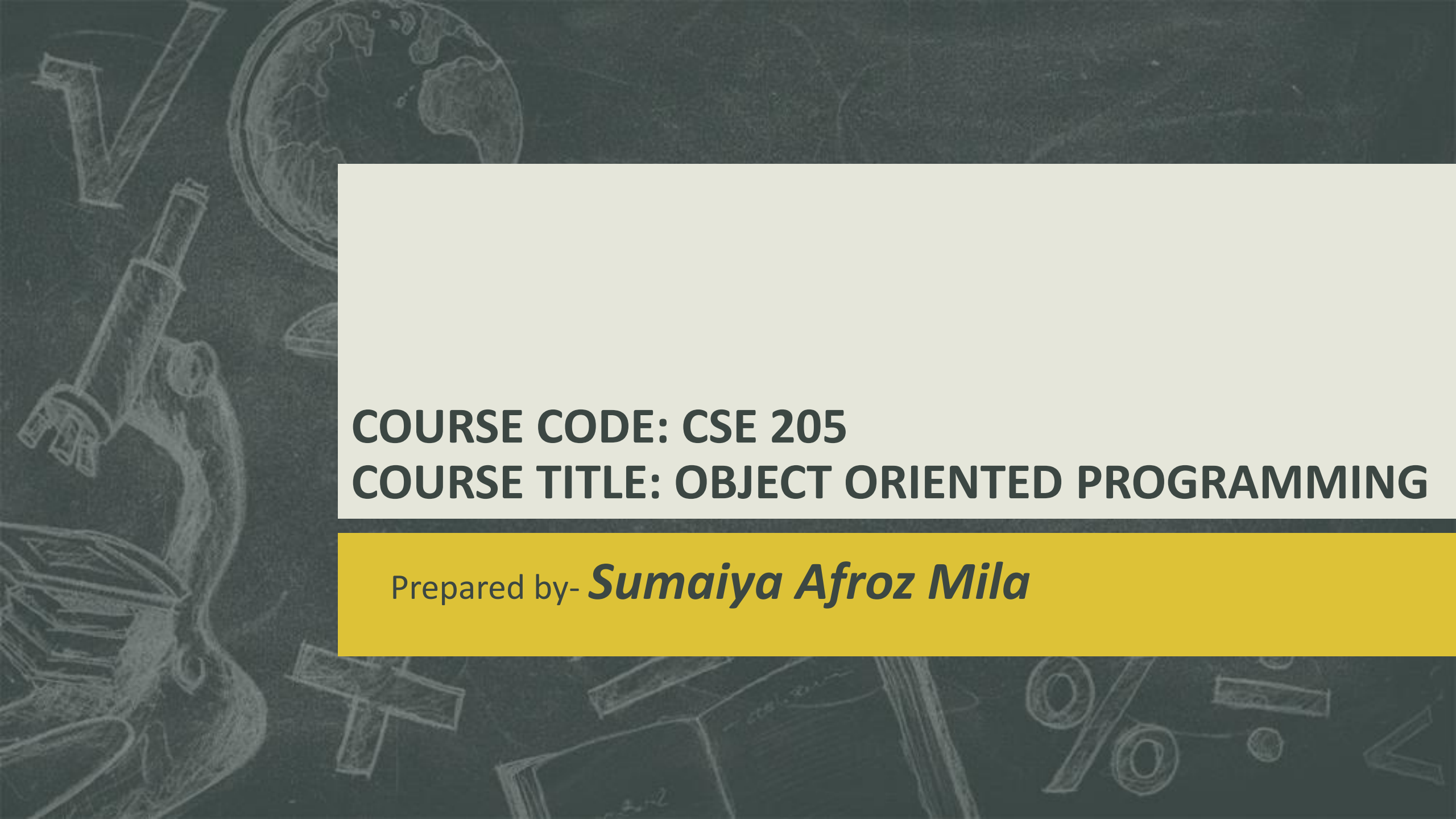


COURSE CODE: CSE 205
COURSE TITLE: OBJECT ORIENTED PROGRAMMING

Prepared by- ***Sumaiya Afroz Mila***

Procedural Programming

How it works

- Split a program into tasks and subtasks
- Write functions to carry out the tasks
- Instruct computer to execute the functions in a sequence

Disadvantages

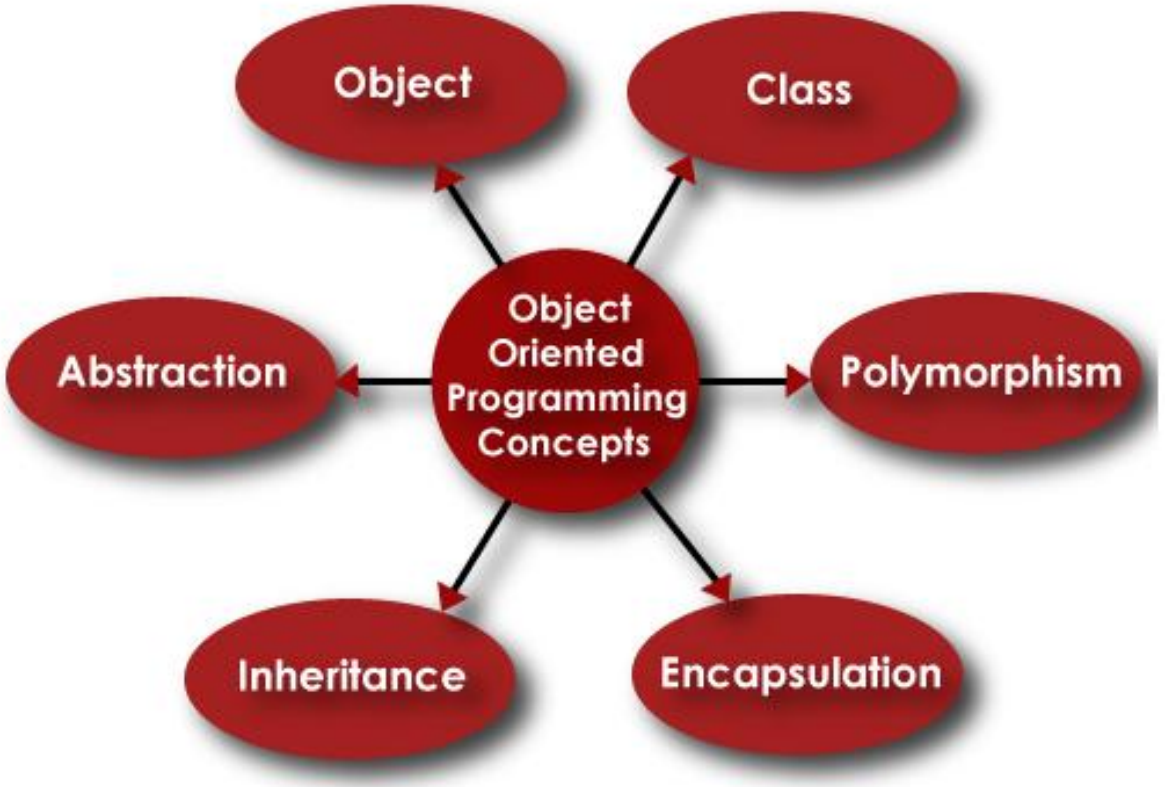
- Large amount of data intricate flow of tasks : complex
- Gives more importance to algorithms & functions than data being used by the functions
- Multi-function programs - shared data items are placed as global
- Changes in data structure affects functions and involves re-writing code

Why OOP?

- OOP treats data as a critical element in the program development.
- Does not allow data to flow freely around the system.
- Supports class/structure creation
- Makes maintenance of large program easy
- Contains following features
    - Encapsulation
    - Polymorphism
    - Inheritance

What is OOP?

Object Oriented programming is a programming style which is associated with the concepts like **class**, object, **Inheritance**, **Encapsulation**, **Abst raction**, **Polymorphism**.



Features of OOP

- Encapsulation
  - Binds together the code and data it manipulates
  - Keeps both safe from accidental interference from outside and misuse
- Polymorphism
  - Allows one name to be used for more than one related but technically different purpose
  - Type of data used to call the function determines which specific version of the function to be executed
  - Function overloading and operator overloading are types of polymorphism
- Inheritance
  - By this process one object can take properties of another

C++ over C

- C++ allows function overloading and operator overloading which reduces redundancy
- Through inheritance, redundant code can be eliminated and extend the use of existing classes
- Principle of data hiding(encapsulation) helps to build a secure program that can not be invaded by code in other parts of the program
- Object oriented systems can be easily upgraded from small to large systems
- Software complexity easily managed

Differences between C & C++

- In C if a function accepts no parameters, then void is mentioned in the parameter list. This is optional in C++

```
void display (void);      // c style
void display () ;         // c++
```

- All funcs must be proto-typed / declared before usage, in C++.
- If a func return type is mentioned it must return a value in C++. In C it is optional.
- In C if return-type of a func is not specified, it is assumed to be int. In C++ it has to be explicitly mentioned. No defaults are allowed
- C++ has an additional bool datatype. C doesnot

C++ Program Structure

- Line 1 & 2 : Preprocessor directives
  - Lines beginning with # are preprocessor commands
  - Include the header file for functions like cin, cout, etc
  - Set the default namespace as standard
- Line 3: Entry point of the program. Begin execution here.
- Line 4-6 : Body of the program

Untitled1.cpp | Untitled1.c

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      cout<<"Hello CSE-19";
6  }
7
```

C vs C++ Output

```
#include <iostream>
using namespace std;
int main(){
cout<<“hello world”;
}
#include <iostream>
using namespace std;
int main(){
int a;
cin>>a;
}
```

```
#include <stdio.h>
int main(){
Printf(“hello world”);
}

#include <stdio.h>
int main(){
int a;
Printf(“%d”,&a);
}
```

C vs C++ Output Cntd.

C++

```
include<iostream>
main(){
int i;
float j;
char c;
cin>> i >> j >> c;
cout<< i <<“ “<<j<<“ “<<c<<endl;
}
```

C

```
include<stdio.h>
main(){
int i;  float j;  char c;
scanf(“%d%f%c”,&i,&j,&c)
;
printf(“%d %f %c\n”,i,j,c);
}
```

Why Using ‘namespace std’?

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

namespace first_space{
    void printf(char *p){
cout << "Inside first_space" << endl; }
 }

int main(){
   printf("hello world");
   first_space::printf("hello world");
}
```

What will be the output?

```
1   #include<iostream>
2   using namespace std;
3
4   void display()
5   {
6       cout<<"Display Function 01"<<endl;
7
8   }
9   void display()
10  {
11      cout<<"Display Function 02"<<endl;
12
13  }
14  int main()
15  {
16      display();
17  }
```

What will be the output???

Error: Void display previously defined here

What is a namespace?

- Additional information to differentiate similar functions, classes, variables etc. with the same name, available in different libraries.
- Namespace, defines a scope. To call namespace-enabled version of either function or variable, prepend the namespace name to the function / variable :

```
std::cout << “hello”<<std::endl;
std::cin.get();
```

- This can be avoided with preprocessor directives :

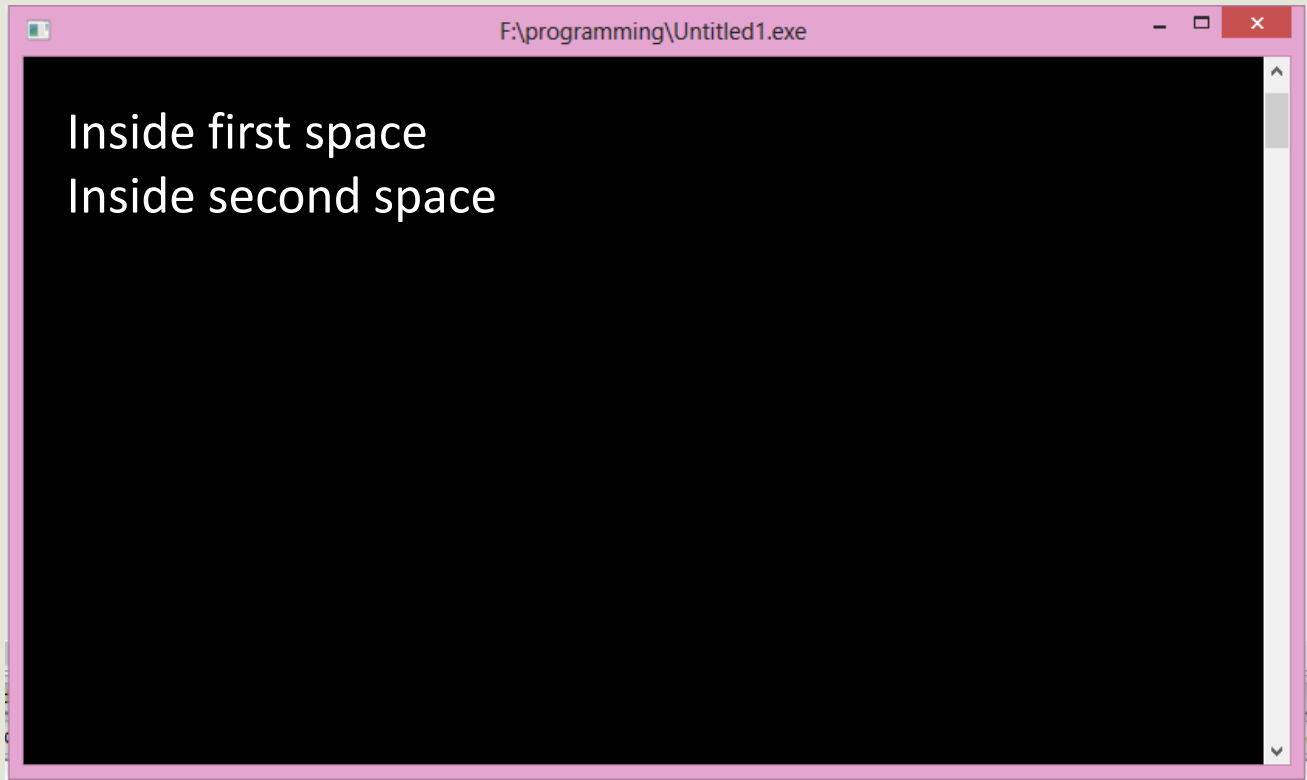
```
using namespace std; // std is abbreviation of namespace standard
cout <<“hello”<<endl;
```

- 3 types of namespace-
  - Standard namespace
  - User defined namespace
  - Anonymous namespace

Using user defined namespace

```
#include <iostream>
using namespace std;
namespace first_space{
    void display(){
cout << "Inside first_space" << endl; }
 }
namespace second_space{
    void display(){
cout << "Inside second_space" << endl; }
 }
int main () {
    // Call function from first name space.
first_space::display();
   // Call function from second name space.
second_space::display();
}
```

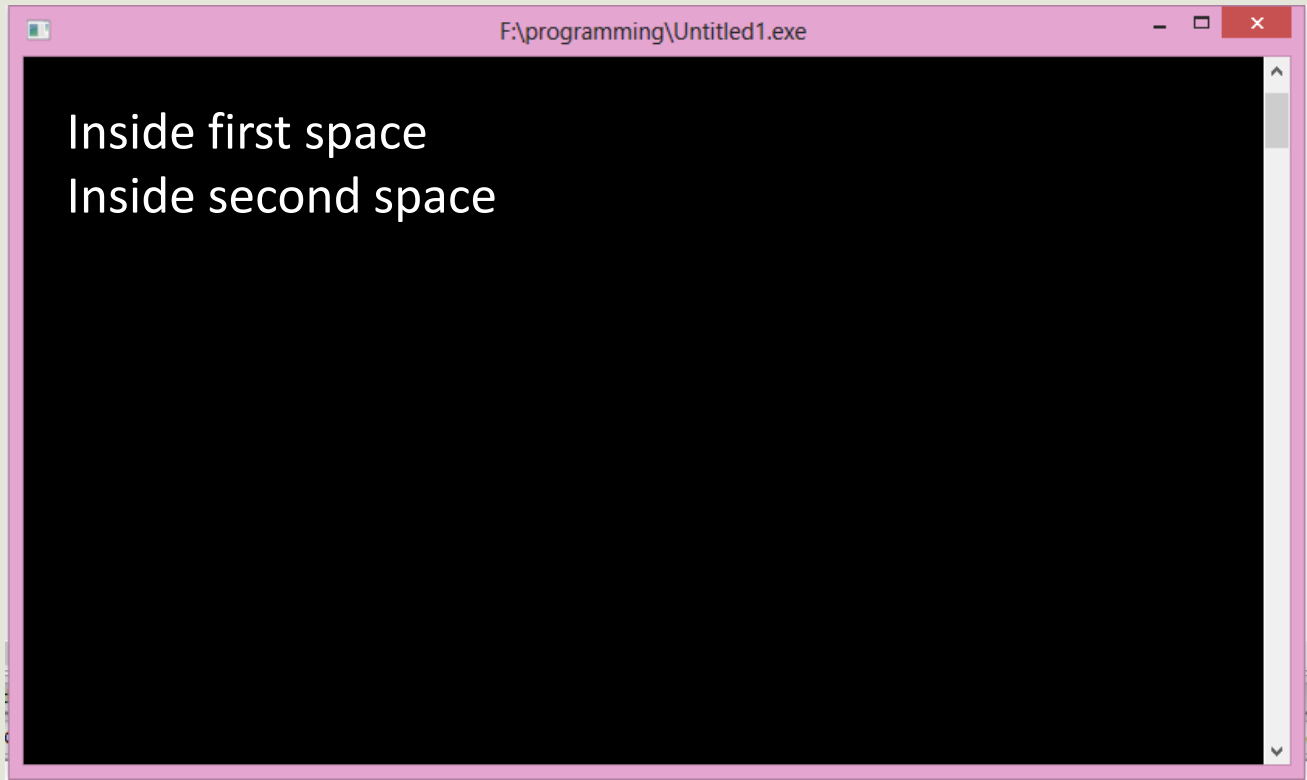
What will be the output?



Using user defined namespace

```
#include <iostream>
using namespace std;
namespace first_space{
    void display(){
cout << “Inside first space”<< endl;
} }
namespace second_space{
    void display(){
cout << “Inside second space”<< endl;
} }
Using namespace first_space ;
int main () {
    // Call function from first name space.
display();
   // Call function from second name space.
second_space::display();
```

What will be the output?



Using user defined namespace

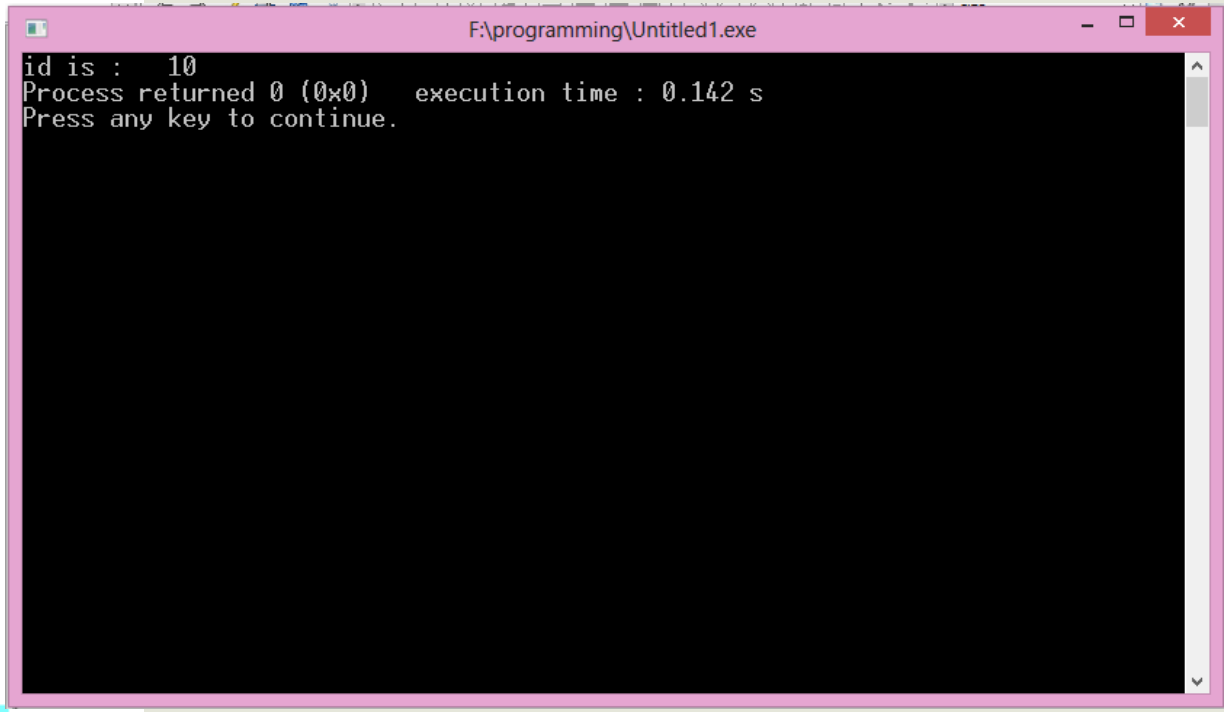
1. **Define two namespaces “SquareSpace” and “CircleSpace” both containing a function with same name “Area”. Use necessary variables.**
2. **Take input of a variable a. Call both functions from main program and show the output. Provide the input as parameter.**
3. **The “Area()” function in the SquareSpace should return the area of a square considering the parameter.**
4. **The “Area()” function in the CircleSpace should return the area of a circle considering the parameter.**

Structure in C

Untitled1.c | Untitled1.cpp

```
1  #include<stdio.h>
2  struct employee{
3      int id;
4      char designation[100];
5      char branch[100];
6  };
7
8  int main(){
9      struct employee emp1;
10     emp1.id = 10;
11     printf("id is :    %d",emp1.id);
12 }
```

What will be the output?



Structure in C

```c
#include<stdio.h>
struct employee{
    int id;
    char designation[100];
    char branch[100];
    int getID(){
        return id;
    }
};
int main(){
    struct employee emp1;
    emp1.id = 10;
    printf("id is :%d",emp1.getID());
}
```

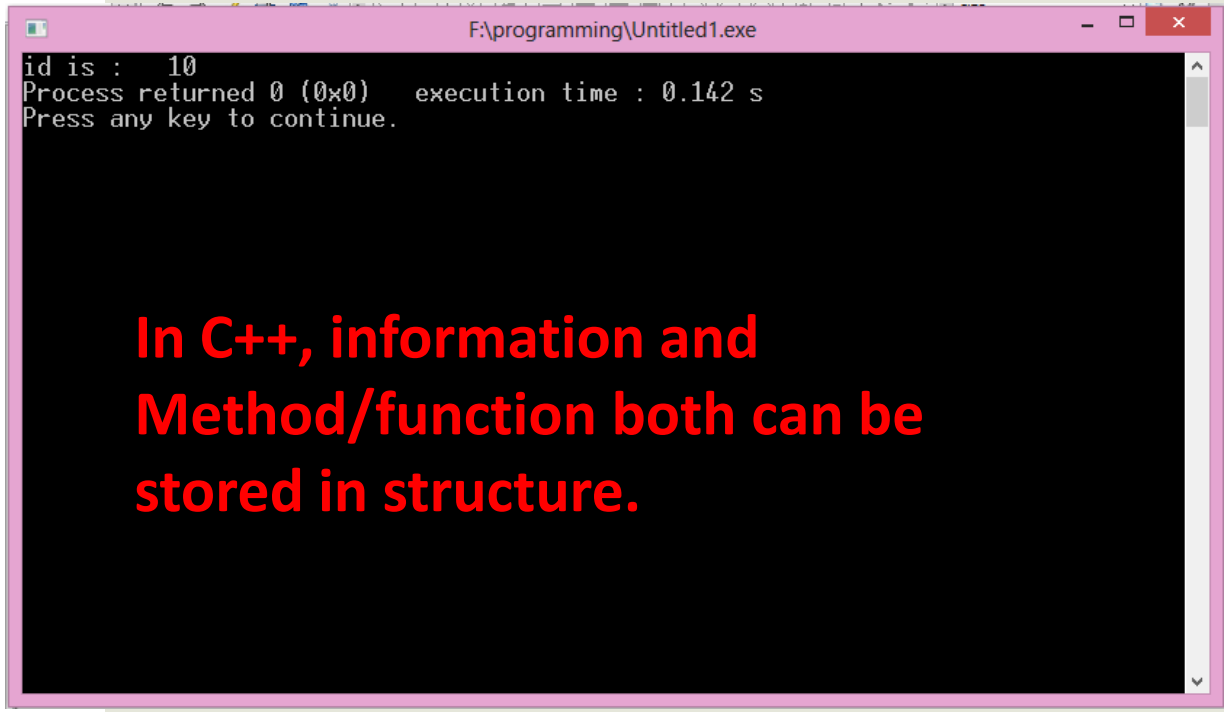
What will happen now?
What will be the output?

There will be compilation error. In C, only information can be stored in structure. Method/function can not be stored.

Structure in C++

```
#include<iostream>
using namespace std;
struct employee{
    int id;
    char designation[100];
    char branch[100];
    int getID(){
        return id;
    }
};
int main(){
    struct employee emp1;
    emp1.id = 10;
    printf("id is :%d",emp1.getID());
}
```

What will happen now?
What will be the output?



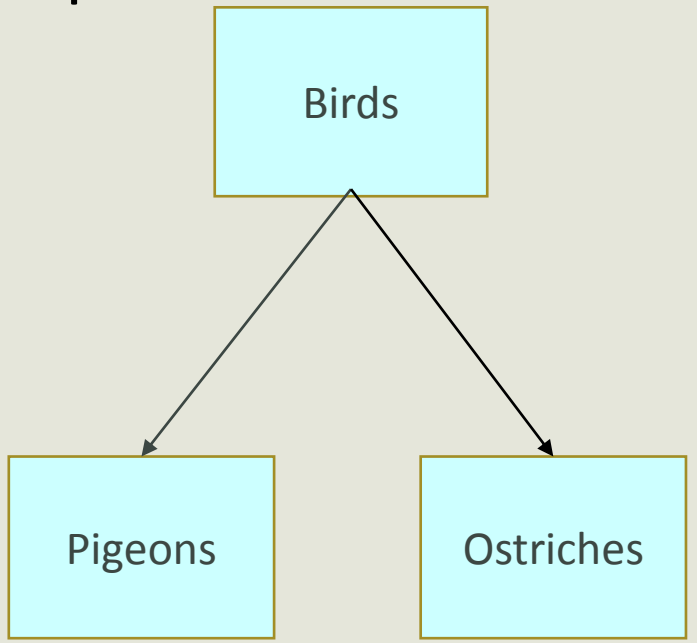
In C++, information and Method/function both can be stored in structure.

Class vs Structure

- Class and structure have identical capabilities
- Both can contain information and methods
- By default members of class are private and members of structure are public.
- Class provides the feature of inheritance and structure does not.

User defined data type: Class

- A class is a collection of method and variables. It is a blueprint that defines the data and behavior of a type.
- Almost all birds have the functionality
  - Have feather
  - Sharp claws
  - Lays eggs
- But birds (pigeons) have some distinct functionality which other birds (ostriches) does not have



- Here we can take Birds as a class. A class is a blueprint for any functional entity which defines its properties and its functions. Like birds having feather, laying eggs.

Objects

- When we say, Birds, Ostriches or Pigeons, we just mean a type/kind
- all the ostriches and pigeons are the forms of these classes. They have a physical existence while a class is just a logical definition. So all the ostriches and pigeons are the objects/instances.

Class and Object more examples

- Assume a **Human** class. Human can talk, walk, eat, see etc.
- Both **Male** and **Female**, performs some common functions, but there are some specifics to both, which is not valid for the other.
- Females can give birth which males can not.
- So **Human**, **Male** and **Female** are classes.
- I am Mila, and I am an **instance** or **object** of **Female** class as well as **Human** class.

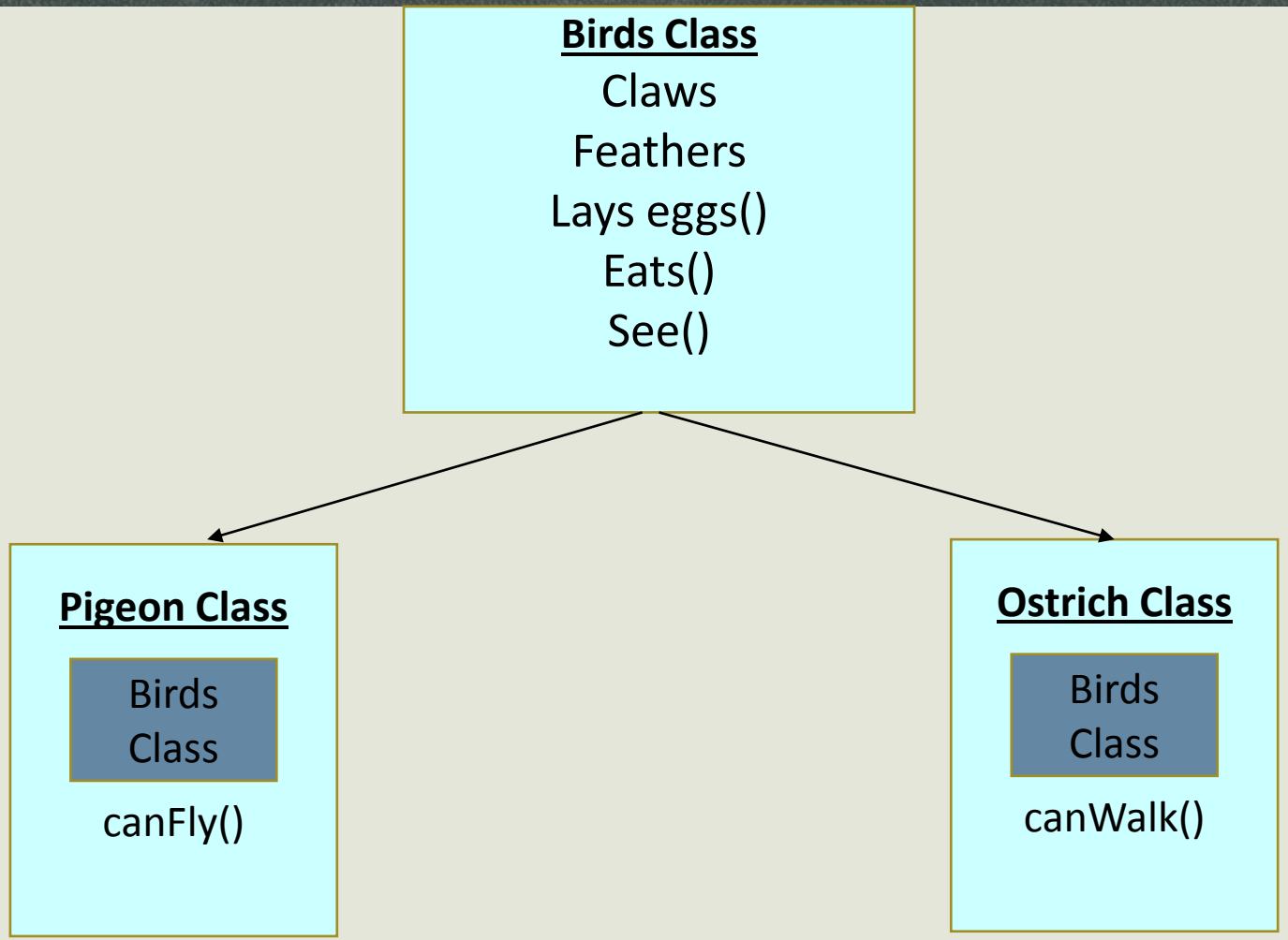
Abstraction

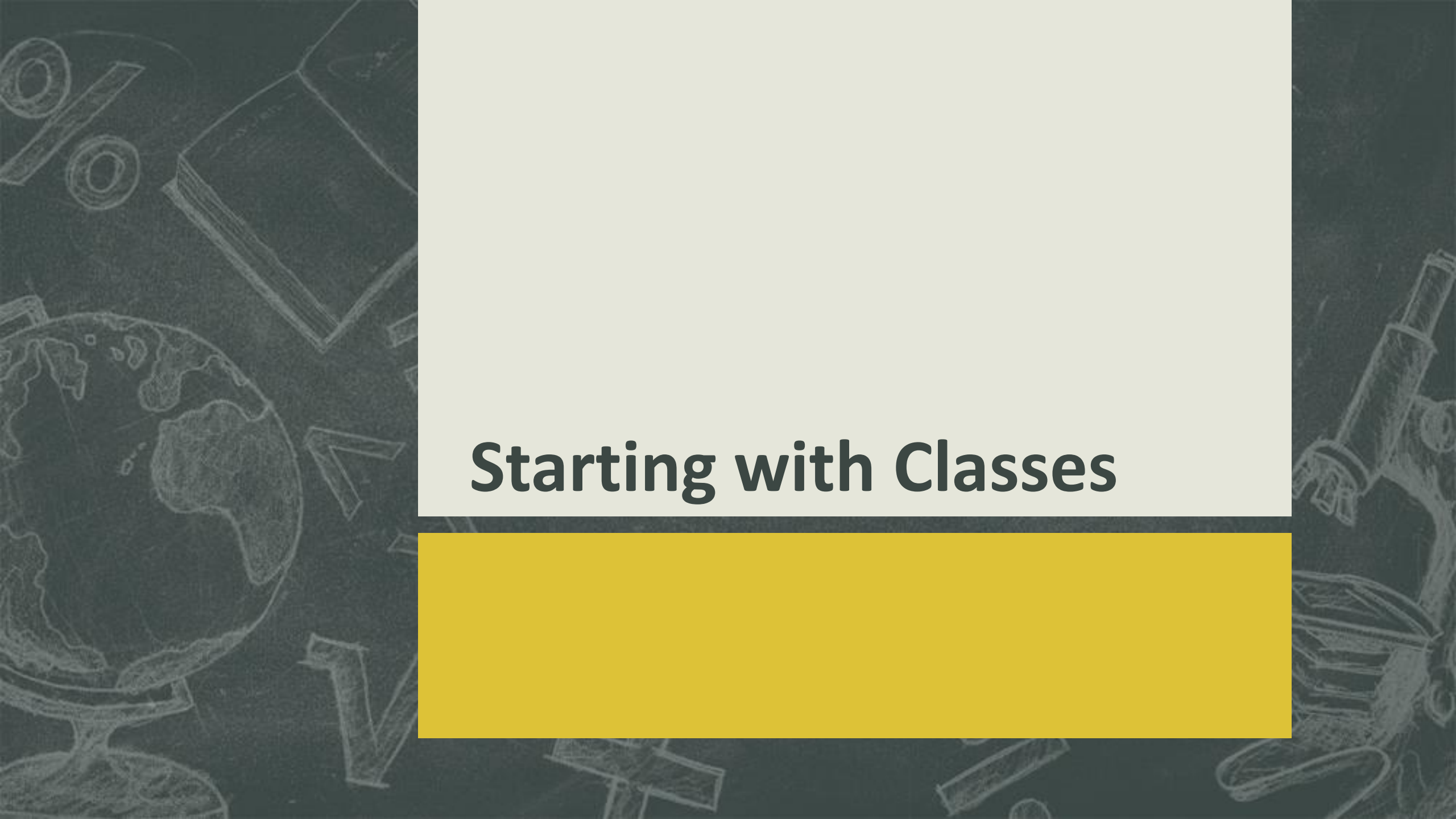
- Abstraction means, showcasing only the required things to the outside world while hiding the details
- Continuing our example, Human can talk, walk, hear, eat, but the details are hidden from the outside world.
- **Difference between abstraction and encapsulation**
  - The most important difference between **Abstraction** and **Encapsulation** is that Abstraction solves the problem at design level while Encapsulation solves it at implementation level.
  - Abstraction is used for hiding the unwanted data and showing the related data. Encapsulation hides the data and code into a single unit to protect the data from outside

Inheritance

- Considering Birds a class, which has properties like claws, feathers, eyes etc, and functions like laying eggs, eat, see etc.
- Chicken and Ostriches are also classes, but most of the properties and functions are included in Birds, hence they can inherit everything from class Birds using the concept of Inheritance.

Inheritance





Starting with Classes

Class

- Class: The building block of C++ that leads to Object Oriented programming is a Class.
- It is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
- A class is like a blueprint for an object/instance.

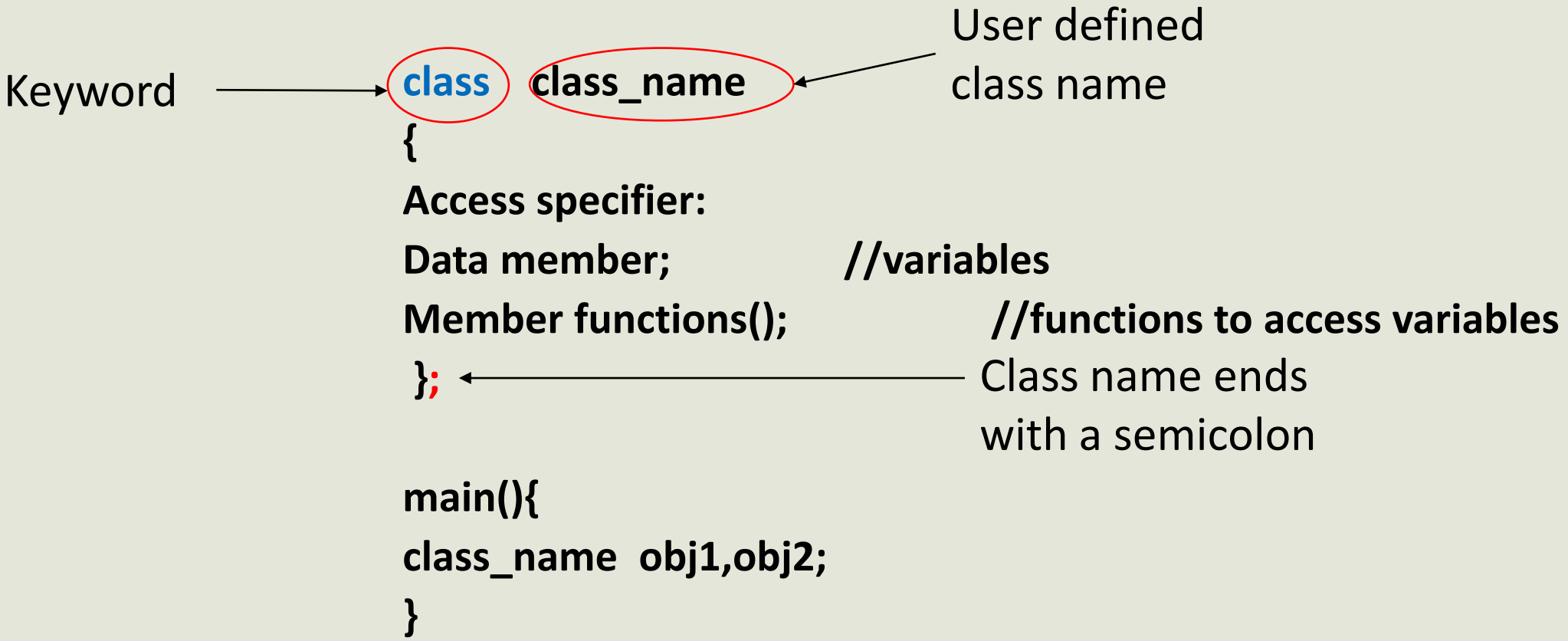
Class

```
struct struct_name{
        //all members
};
main(){
struct struct_name st1,st2;
 }
```

```
class class_name{
//private members
public:
 //public members
 };

main(){
class_name  obj1,obj2;
}
```

Class


Keyword
class
class_name
User defined
class name
{
Access specifier:
Data member; //variables
Member functions(); //functions to access variables
};
Class name ends
with a semicolon
main(){
class_name obj1,obj2;
}

You should also study these topics…

- Differences between C and C++ (teach yourself 1.6)
- What is abstraction and what is encapsulation? Explain with example.(https://www.guru99.com/difference-between-abstraction-and-encapsulation.html)


Thank
you!