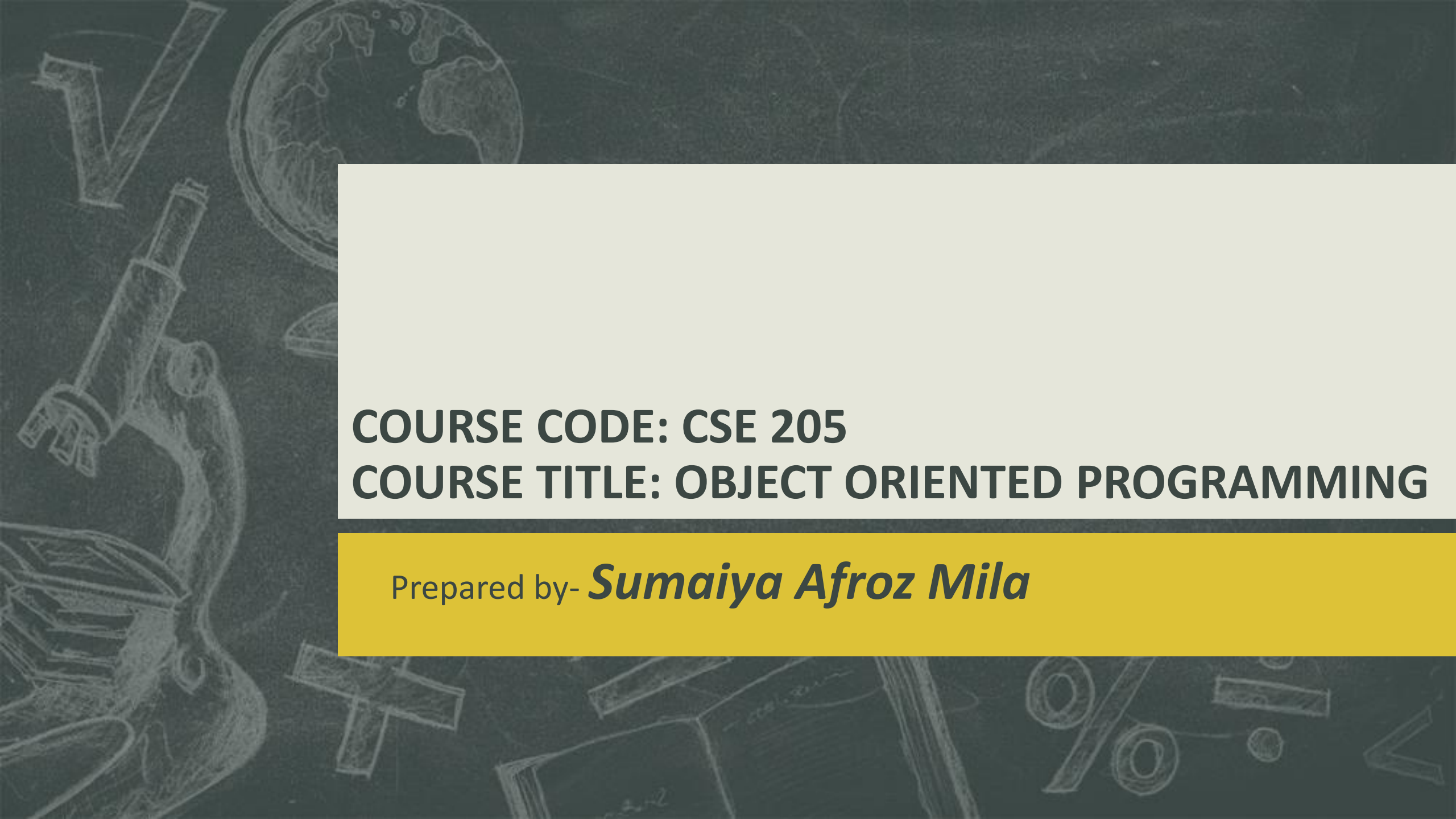


COURSE CODE: CSE 205
COURSE TITLE: OBJECT ORIENTED PROGRAMMING

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

Static Class Member

- When a member variable is declared as static, It causes only **one copy of that variable to exist – no matter how many objects of that class are declared.**
- All the objects of that class simply **shares** that variable.
- Same static variable will be used by any classes derived from the class that contains a static variable – related to inheritance
- A static member variable exists ***before*** any object of that class is created.
- A static class member is a global variable that simply has its scope restricted to the class in which it is declared.

Static Class Member

- A static variable can only be declared inside a class. Definition must be outside the class.
- All static member variables are initialized to 0 **by default**.
- However, it is possible to choose an initial value other than 0 for the static member variable.
- A member function of a class can be declared as static

Assignment

- Why is it necessary to use static data members?
- Why static member function does not have “this” pointer?
- Why non static member variable can not be accessed by a static function?

Example 1

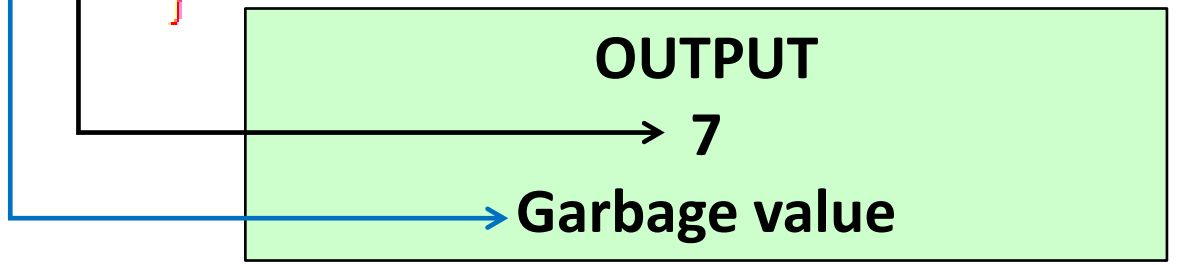
```
class myclass
{
    int x;
public:
    void setx(int n)
    {x = n; }
    int getx()
    {return x; }
};
```

```
int main()
{
myclass o1, o2;
o1.setx(7);
cout<< o1.getx() <<endl;
cout<< o2.getx() <<endl;
}
```

Example 1

```
class myclass
{
    int x;
public:
    void setx(int n)
    {x = n; }
    int getx()
    {return x; }
};
```

```
int main()
{
myclass o1, o2;
o1.setx(7);
cout<< o1.getx() <<endl;
cout<< o2.getx() <<endl;
}
```



Example 2

```
class myclass
{
      static int x;
public:
      void setx(int n)
      {x = n; }
      int getx()
      {return x; }
};

int myclass::x;
```

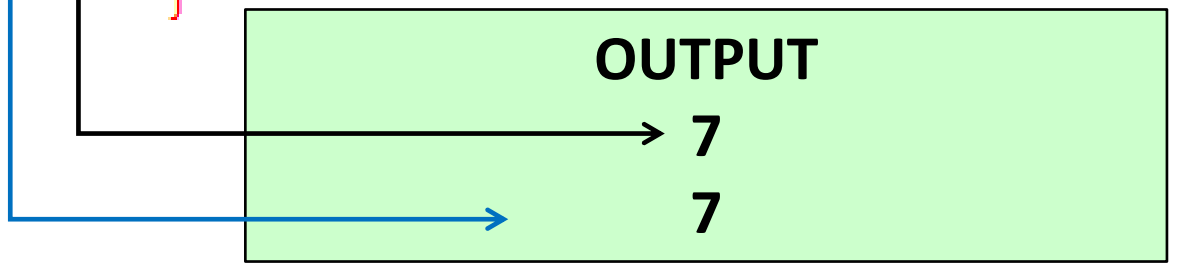
```
int main()
{
myclass o1, o2;
o1.setx(7);
cout<< o1.getx() <<endl;
cout<< o2.getx() <<endl;
}
```

Example 2

```
class myclass
{
      static int x;
public:
      void setx(int n)
      {x = n; }
      int getx()
      {return x; }
};

int myclass::x;
```

```
int main()
{
myclass o1, o2;
o1.setx(7);
cout<< o1.getx() <<endl;
cout<< o2.getx() <<endl;
}
```



Example 3

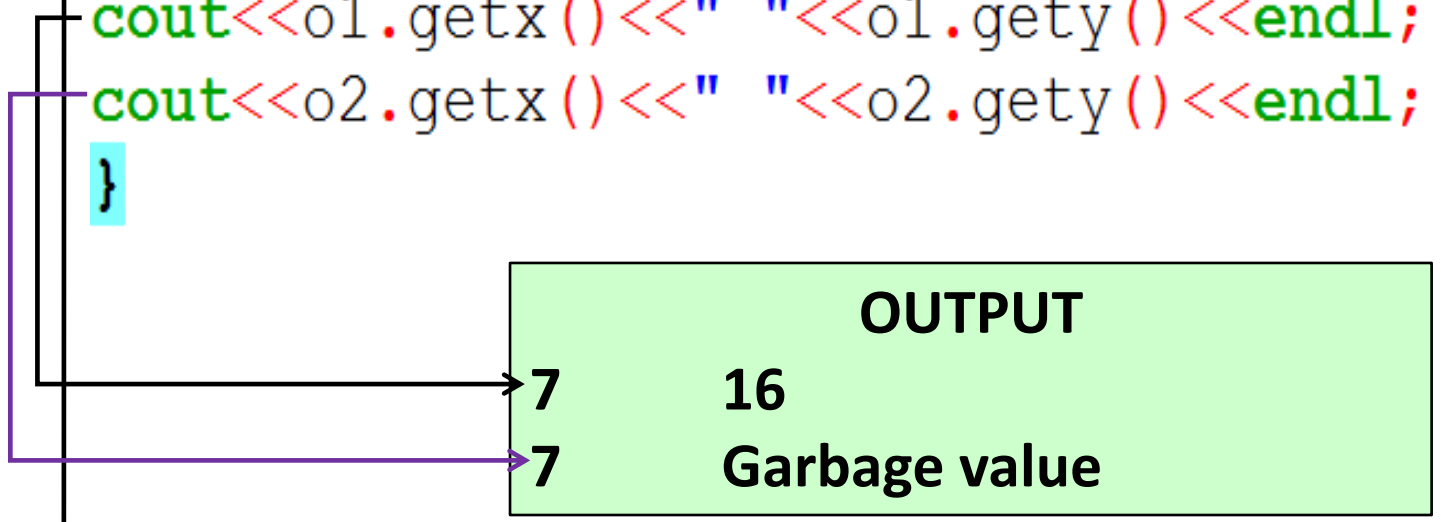
```
class myclass{
    static int x;
    int y;
public:
    void setx(int n){
        x = n; }
    void sety(int n){
        y = n; }
    int getx(){
        return x; }
    int gety(){
        return y; }
};
int myclass::x;
```

```
int main()
{
myclass o1, o2;
o1.setx(7);
o1.sety(16);
cout<<o1.getx()<<" "<<o1.gety()<<endl;
cout<<o2.getx()<<" "<<o2.gety()<<endl;
}
```

Example 3

```
class myclass{
    static int x;
    int y;
public:
    void setx(int n){
        x = n; }
    void sety(int n){
        y = n; }
    int getx(){
        return x; }
    int gety(){
        return y; }
};
int myclass::x;
```

```
int main()
{
myclass o1, o2;
o1.setx(7);
o1.sety(16);
cout<<o1.getx()<<" "<<o1.gety()<<endl;
cout<<o2.getx()<<" "<<o2.gety()<<endl;
}
```



Example 4

```
class myclass{
public:
    static int x;
    int y;
    void setx(int n){
        x = n; }
    void sety(int n){
        y = n; }
    int getx(){
        return x; }
    int gety(){
        return y; }
};
int myclass::x;
```

```
int main()
{
myclass o1, o2;
myclass :: x = 7;
o1.sety(10);
o2.sety(16);
cout<<o1.getx()<<" "<<o1.gety()<<endl;
cout<<o2.getx()<<" "<<o2.gety()<<endl;
}
```

Example 4

```
class myclass{
public:
    static int x;
    int y;
    void setx(int n){
        x = n; }
    void sety(int n){
        y = n; }
    int getx(){
        return x; }
    int gety(){
        return y; }
};
int myclass::x;
```

```
int main()
{
myclass o1, o2;
myclass :: x = 7;
o1.sety(10);
o2.sety(16);
cout<<o1.getx()<<" "<<o1.gety()<<endl;
cout<<o2.getx()<<" "<<o2.gety()<<endl;
}
```

OUTPUT

7 10

7 16

Example 6

```
class myclass{
public:
    static int x;
    int y;
    void setx(int n){
        x = n; }
    void sety(int n){
        y = n; }
    int getx(){
        return x; }
    int gety(){
        return y; }
};
int myclass::x;
```

```
int main()
{
myclass::x=7;
myclass o1, o2;
o1.sety(10);
o2.sety(16);
cout<<o1.getx()<<" "<<o1.gety()<<endl;
cout<<o2.getx()<<" "<<o2.gety()<<endl;
}
```

It is possible to use a static variable before creating any object of the class

Example 5

```
class myclass{
private:
    static int x;
    int y;
public:
    void setx(int n){
        x = n; }
    void sety(int n){
        y = n; }
    int getx(){
        return x; }
    int gety(){
        return y; }
};
int myclass::x  ;
```

```
int main()
{
myclass o1, o2;
myclass :: x = 7;
o1.sety(10);
o2.sety(16);
cout<<o1.getx()<<" "<<o1.gety()<<endl;
cout<<o2.getx()<<" "<<o2.gety()<<endl;
}
```

Compilation error

int myclass::x is private

Example 5

```
15          return x; }
16      int gety(){
17          return y; }
18  };
19  int myclass::x;
20
21  int main()
22  {
23  myclass o1, o2;
24  myclass :: x = 7;
25  o1.sety(10);
26  o2.sety(16);
27  cout<<o1.getx()<<"
```

Blocks | Search results | Cccc | Build log

	Line	Message
mming...	19	error: 'int myclass::x' is private
mming...	24	error: within this context

Example 5

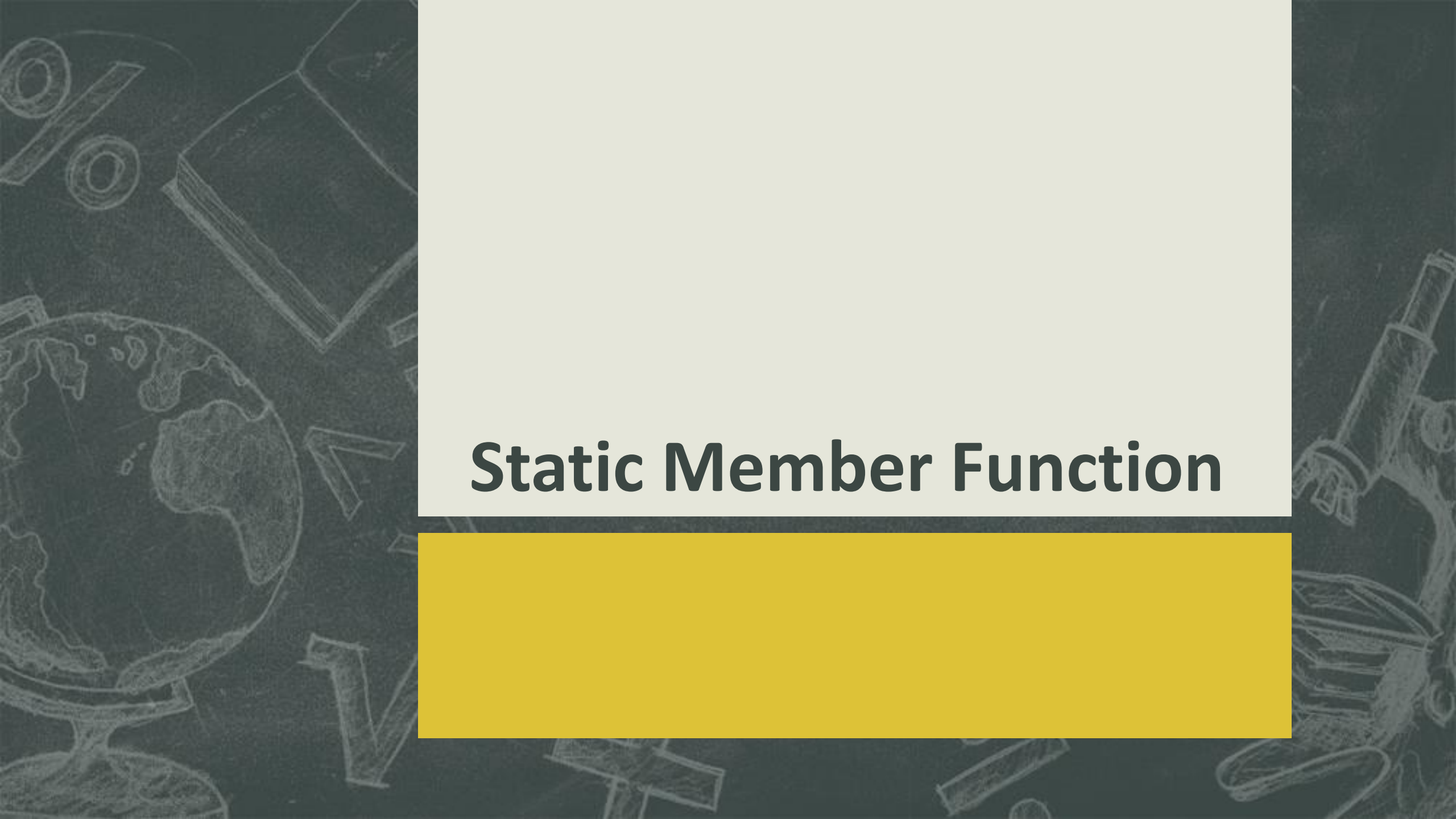
```
class myclass{
private:
     static int x;
     int y;
public:
     void setx(int n){
          x = n; }
     void sety(int n){
          y = n; }
     int getx(){
          return x; }
     int gety(){
          return y; }
};
int myclass::x=8;
```

```
int main()
{
myclass o1, o2;

o1.sety(10);
o2.sety(16);
cout<<o1.getx()<<" "<<o1.gety()<<endl;
cout<<o2.getx()<<" "<<o2.gety()<<endl;
}
```

Initializing static variable within class is not allowed

Still, you can initialize here.



Static Member Function

Example 1

```
class myclass{
public:
    static void show(int n){
    cout<<"Value of n: "<<n<<endl;
    }
};
```

As static member functions are not attached to a particular object, they can be called directly by using the class name and the scope resolution operator.

```
int main()
{
myclass ob1;
ob1.show(10);
myclass::show(15);
}
```

OUTPUT

Value of n: 10

Value of n: 15

Example 2

```
class myclass{
public:
    static void show(int n){
    cout<<"Value of n: "<<n<<endl;
    }
};
```

It is possible to use a static function *before* any object instantiation

```
int main()
{
myclass::show(15);
myclass ob1;
ob1.show(10);
}
```

OUTPUT

Value of n: 15

Value of n: 10

Example 3

```
class myclass{
public:
    static int x;
    static void setx(int n){
        x=n;
    }
    int getx(){
        return x;
    }
};
int myclass::x;
```

```
int main()
{
myclass::setx(15);
myclass ob1;
cout<<ob1.getx()<<endl;
ob1.setx(10);
cout<<ob1.getx()<<endl;
}
```

It is possible to use a static variable/function or global variable/ function inside a static function

OUTPUT

Value of n: 15

Value of n: 10

Example 3

```
class myclass{
public:
        int x;
    static void setx(int n){
        x=n;
    }
    int getx(){
        return x;
    }
};
int myclass::x;
```

What will happen if non static member variable is accessed inside static function??

```
int main()
{
myclass::setx(15);
myclass ob1;
cout<<ob1.getx()<<endl;
ob1.setx(10);
cout<<ob1.getx()<<endl;
}
```

OUTPUT

Value of n: 15

Value of n: 10

Static Member Function

- A static member function can access only other static members of its class.
- It can access non-static global data and functions
- Static member function does not have a “this” pointer
- Virtual static member functions are not allowed - inheritance
- A static member function can be invoked by an object of its class, or it can be called independent of any object. Via the class name and the scope resolution operator.


Thank
you!