static members

Static data members are class members that are declared using the static keyword. There is only one copy of the static data member in the class, even if there are many class objects. This is because all the objects share the static data member. The static data member is always initialized to zero when the first object of class is created. Static members are initialized before main() and are always destroyed when program is terminated.

Example1: access static variable

```
class my_Class {
   public:
      static int num;
};
int my_Class :: num=25;

int main(void) {
   cout << my_Class :: num ;
}
```

Example2: access static variable with object

```
class my_Class {
   public:
      static int num;
};
int my_Class :: num=25;

int main(void) {
   my_Class obj;
   cout << obj.num ;
}
```

static member function()

Static member functions in C++ are the functions that can access only the static data members. These static data members share a single copy of themselves with the different objects of the same class. A function can be made static by using the keyword static before the function name while defining a class.
Static members functions() are initialized before main() and are always destroyed when program is terminated.

Syntax:
static returntype function_name() {}

Example1: access static variable with static method()

```
class my_Class {
    static int num;
    public:
        static void static_method() {
            cout << num << endl;
        }
};
int my_Class::num=25;
int main(void) {
    my_Class obj;
    obj.static_method();
}
```

Example2: access non-static variable with static method()

```
class my_Class {
    int num;
    public:
        static void static_method() {
            cout << num;
        }
};
int main(void) {
    my_Class obj;
    obj.static_method();
}
```

Note: you cannot access non-static member in static method().

Example3: access static and non-static variable with non-static method()

```
class my_Class {
    static int num1;
    int num2=10;
    public:
        void non_static_method() {
            cout << num1 << " " <<num2;
        }
};
int my_Class :: num1=25;

int main(void) {
    my_Class obj;
    obj.non_static_method();
}
```

Example4: Student Record Management

```
class my_Class {
    public:
        static int Roll_No;
        string Name;
        long long int Contact;
        int Age;
        float Percentage;

        void insert_Data(string Name, long long int Contact, int Age, float 
Percentage){
            this-> Name = Name;
            this-> Contact = Contact;
            this->Age = Age;
            this->Percentage = Percentage;
        }
        void print_Data(){
            cout << "\t" <<++Roll_No << "\t" << Age << "\t" << Name << "\t" << 
Contact << "\t" << Percentage <<endl;
        }
};
int my_Class :: Roll_No=0;
int main(void) {
    my_Class obj1,obj2,obj3,obj4;
    cout << "\tRoll \tAge \tName \t\tContact \tPercentage" << endl;
    obj1.insert_Data("Student1",1234567890, 17, 90.7458);
    obj1.print_Data();
    obj2.insert_Data("Student2",2345678901, 18, 67.796);
    obj2.print_Data();
    obj3.insert_Data("Student3",3456789012, 19, 85.2457);
    obj3.print_Data();
}
```

How to Print Decimal values,

```
printf();         // C Style
floor();          // roundoff the value       <cmath>
ceil();           // roundon the value        <cmath>
trunc();          // removes decimal digit  <cmath>
round();          // rounds the value         <cmath>
setprecision();        // set number of decimal points           <iomanip>
```

Example:

```
int main(void) {
    float temp1 = 100.22, temp2 = 100.67;
    cout << "Entered Values = " << temp1 << "\t" << temp2 << endl<<endl;

    printf("C Style > %.2f \t %.5f \n",temp1,temp2);
    cout << "floor() > temp1 = " << floor(temp1) << "\t" << "temp2 = " <<
floor(temp2) <<endl;

    cout << "ceil() > temp1 = " << ceil(temp1) << "\t" << "temp2 = " <<
ceil(temp2) <<endl;

    cout << "trunc() > temp1 = " << trunc(temp1) << "\t" << "temp2 = " <<
trunc(temp2) <<endl;

    cout << "round() > temp1 = " << round(temp1) << "\t" << "temp2 = " <<
round(temp2) <<endl;

    cout << fixed << setprecision(0) <<"trunc() > temp1 = " << temp1  << "\t" <<
setprecision(5) << "temp2 = " << temp2 <<endl;
}
```

static objects

An object becomes static when a static keyword is used in its declaration. Static objects are initialized only once and live until the program terminates.

Example:

```
class my_Class{
   public:
      print(){
         cout << "Class executed with static object";
      }
};

void myfunc()
{
   static my_Class obj;
   obj.print();
}

int main(void)
{
   myfunc();
}
```