};

Objects as Function Arguments:

```
// C++ program to calculate the average marks of two students

#include <iostream>
using namespace std;
class Student {
  public:
        double marks;
        // constructor to initialize marks
        Student(double m) {
        marks = m;
        }
};

// function that has objects as parameters
void calculateAverage(Student s1, Student s2) {

        // calculate the average of marks of s1 and s2
        double average = (s1.marks + s2.marks) / 2;

  cout << "Average Marks = " << average << endl;

}

int main() {
        Student student1(90.0), student2(90.0);

 // pass the objects as arguments
  calculateAverage(student1, student2);

        return 0;
}
```

Returning Objects from functions:

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

class Student {
  public:
        double marks1, marks2;
};
```

```
// function that returns object of Student
Student createStudent() {
        Student student;

        // Initialize member variables of Student
        student.marks1 = 96.5;
        student.marks2 = 75.0;

        // print member variables of Student
        cout << "Marks 1 = " << student.marks1 << endl;
        cout << "Marks 2 = " << student.marks2 << endl;

        return student;
}

int main() {
        Student student1;

        // Call function
        student1 = createStudent();

        return 0;
}
```

In this program, we have created a function createStudent() that returns an object of Student class.

We have called createStudent() from the main() method.

```
// Call function
student1 = createStudent();
```

Here, we are storing the object returned by the createStudent() method in the student1.

Differences between structure and class:

Here, we are going to discuss the main differences between the structure and class. Some of them are as follows:

- By default, all the members of the structure are public. In contrast, all members of the class are private.
- The structure will automatically initialize its members. In contrast, constructors and destructors are used to initialize the class members.

- When a structure is implemented, memory allocates on a stack. In contrast, memory is allocated on the heap in class.
- Variables in a structure cannot be initialized during the declaration, but they can be done in a class.
- There can be no null values in any structure member. On the other hand, the class variables may have null values.
- A structure is a value type, while a class is a reference type.
- Operators to work on the new data form can be described using a special method.

Static Data members:

Static data members are class members that are declared using static keywords. A static member has certain special characteristics. These are:

- Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.
- It is initialized before any object of this class is being created, even before main starts.
- It is visible only within the class, but its lifetime is the entire program

Syntax

static data_type data_member_name;

program to access the static member function using the class name in the C++ programming language.

```
#include <iostream>
using namespace std;
class Note
{
// declare a static data member
static int num;

public:
// create static member function
static int func ()
{
return num;
}
};
// initialize the static data member using the class name and the scope resolution operator
int Note :: num = 5;

int main ()
{
// access static member function using the class name and the scope resolution
cout << " The value of the num is: " << Note:: func () << endl;
return 0;
}
```

program to access the static member function using the class' object in the C++ programming language.

```
#include <iostream>
using namespace std;
class Note
{
// declare a static data member
static int num;

public:
// create static member function
static int func ()
{
cout << " The value of the num is: " << num << endl;
}
};
// initialize the static data member using the class name and the scope resolution operator
int Note :: num = 15;

int main ()
{
   // create an object of the class Note
   Note n;
// access static member function using the object
n.func();

return 0;
}
```

access the static member function using the object and class in the C++ programming language.

```
#include <iostream>
using namespace std;
class Member
{

private:
// declaration of the static data members
static int A;
static int B;
static int C;

// declare public access specifier
public:
// define the static member function
static void disp ()
{
cout << " The value of the A is: " << A << endl;
cout << " The value of the B is: " << B << endl;
cout << " The value of the C is: " << C << endl;
}
};
// initialization of the static data members
int Member :: A = 20;
int Member :: B = 30;
int Member :: C = 40;

int main ()
{
// create object of the class Member
Member mb;
// access the static member function using the class object name
cout << " Print the static member through object name: " << endl;
mb. disp();
// access the static member function using the class name
cout << " Print the static member through the class name: " << endl;
Member::disp();
return 0;
}
```