

# Friend Classes & Member Friend Functions in C++ | C++ Tutorials for Beginners #27

In this tutorial, we will discuss friend classes and member friend functions in C++

## Member Friend Functions in C++

Friend functions are those functions that have the access to private members of the class in which they are declared. The main thing to note here is that only that function can access the member function which is made a friend of the other class. An example of the friend function is shown below.

```
class Complex
{
    int a, b;

    // Individually declaring functions as friends
    friend int Calculator ::sumRealComplex(Complex, Complex);
    friend int Calculator ::sumCompComplex(Complex, Complex);

public:
    void setNumber(int n1, int n2)
    {
        a = n1;
        b = n2;
    }

    void printNumber()
    {
        cout << "Your number is " << a << " + " << b << "i" << endl;
```

```

    }
};

int Calculator ::sumRealComplex(Complex o1, Complex o2)
{
    return (o1.a + o2.a);
}

int Calculator ::sumCompComplex(Complex o1, Complex o2)
{
    return (o1.b + o2.b);
}

```

Copy

### Code Snippet 1: Friend function example

As shown in a code snippet 1, a complex class is created which consists of two friend functions “sumRealComplex” and “sumCompComplex” of the calculator class. The main thing to note here is that “sumRealComplex” and “sumCompComplex” are the friend functions of complex class so they can access all the private members of the complex class.

### Friend Classes in C++

Friend classes are those classes that have permission to access private members of the class in which they are declared. The main thing to note here is that if the class is made friend of another class then it can access all the private members of that class. An example program to demonstrate friend classes in C++ is shown below.

```

// Forward declaration
class Complex;

class Calculator
{

```

```

public:
    int add(int a, int b)
    {
        return (a + b);
    }

    int sumRealComplex(Complex, Complex);
    int sumCompComplex(Complex, Complex);
};

```

Copy

### Code Snippet 2: Calculator Class

As shown in code snippet 2, a complex class is declared at the top which is known as forward declaration. Forward declaration hints to the compiler that this class is declared somewhere forward in the code. After that calculator class is defined this consists of three public member functions, "add", "sumRealComplex", and "sumCompComplex". The "add" function will add the values of "a" and "b" and return the value. The "sumRealComplex" and "sumCompComplex" are taking two objects of the complex class. The code for the complex class is shown below.

```

class Complex
{
    int a, b;

    // Individually declaring functions as friends
    // friend int Calculator ::sumRealComplex(Complex, Complex);
    // friend int Calculator ::sumCompComplex(Complex, Complex);

    // Aliter: Declaring the entire calculator class as friend
    friend class Calculator;

public:
    void setNumber(int n1, int n2)

```

```

    {
        a = n1;
        b = n2;
    }

    void printNumber()
    {
        cout << "Your number is " << a << " + " << b << "i" << endl;
    }
};

int Calculator ::sumRealComplex(Complex o1, Complex o2)
{
    return (o1.a + o2.a);
}

int Calculator ::sumCompComplex(Complex o1, Complex o2)
{
    return (o1.b + o2.b);
}

```

Copy

### Code Snippet 3: Complex Class

As shown in code snippet 3, a complex class is defined which consists of, two private data members "a" and "b", and two public member functions "setNumber" and "printNumber". The function "setNumber" will assign the values to the variables "a" and "b". The function "printNumber" will print the values of the variables "a" and "b". Two functions "sumRealComplex" and "sumCompComplex" are defined at the end. The function "sumRealComplex" will add the real values and the function "sumCompComplex" will add the complex value. The main program is shown below.

```

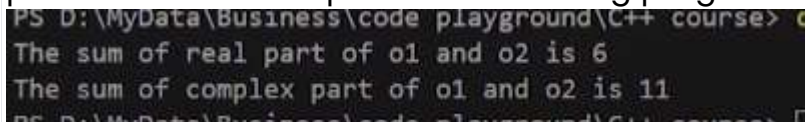
int main()
{
    Complex o1, o2;
    o1.setNumber(1, 4);
    o2.setNumber(5, 7);
    Calculator calc;
    int res = calc.sumRealComplex(o1, o2);
    cout << "The sum of real part of o1 and o2 is " << res << endl;
    int resc = calc.sumCompComplex(o1, o2);
    cout << "The sum of complex part of o1 and o2 is " << resc <<
endl;
    return 0;
}

```

Copy

#### Code snippet 4: Main Program

As shown in code snippet 4, 1<sup>st</sup> two objects "o1" and "o2" of the "complex" data type are declared. 2<sup>nd</sup> "setNumber" function is called with the "o1" and "o2" objects and the values are passed. 3<sup>rd</sup> object "calc" of the calculator data type is declared. 4<sup>th</sup> "sumRealComplex" function is called by the "calc" object and the object "o1" and "o2" are passed to it. 5<sup>th</sup> "sumCompComplex" function is called by the "calc" object and the object "o1" and "o2" are passed to it. The output of the following program is shown in figure 1.



```

PS D:\MyData\Business\code playground\C++ course> c
The sum of real part of o1 and o2 is 6
The sum of complex part of o1 and o2 is 11
PS D:\MyData\Business\code playground\C++ course>

```

#### Figure 1: Program Output

As shown in figure 1, the sum of the real part is shown which is "6" and the sum of the complex part is shown which is "11".