Lab 3: Introduction to classes and objects

Imran Ali

2024-09-10

- In this lab we will write object oriented code for implementing fraction arithmetic.
- We will write many versions of the code to understand the purpose of various object oriented features.
- Write the following code, compile and run it:

```
#include<iostream>
using namespace std;
class Fraction {
    // member functions
    public:
    void setValues(int num, int denom) {
        numerator = num;
        denominator = denom;
    }

    void print() {
        cout << numerator << "/" << denominator << endl;
    }

    // data members
    int numerator;
    int denominator;
};
int main()
{
 Fraction f1;
 f1.setValues(1,2);
 f1.print();
 f1.numerator=2;
 f1.denominator=5;
 f1.print();
 return 0;
}
```

- Notice that the `numerator` and `denominator` are accessible in `main` function.
- It is because every member in the class is `public`

Task1:

- Add the line `private:` before the line `int numerator;`
- Find why the program does not compile?
- Comment following lines in the code:

```
f1.numerator=2;
f1.denominator=5;
f1.print();
```

- Now try to compile and run your code.

Task2:

- In `main` create another fraction `f2`
- call the `setValue` function on `f2` passing two values 2 and 5
- call the `print` function on `f2`

Constructor

- Constructor is a special function which is automatically called when you create an object.
- The job of a constructor is to initialize **data members**.
- It has the same name as that of the class.
- It has no return type.
- Now add the following constructor to your code:

```

class Fraction {
    // member functions
    public:
    Fraction()
    {
     numerator = 2;
     denominator = 5;
    }
    // remaining code in the class remains the same
};
```

- To see how this constructor works, change the `main` function as follows:

```
int main()
{
 Fraction f1;
 f1.print();
 return 0;
}
```

More than one constructors

- It is possible to create more than one constructors in a class.
- Providing more than one type of constructors allow to create object in different ways.
- Add the following constructor to the class:

```
Fraction(int n, int d)
{
   numerator = n;
   denominator = d;
}
```

- Modify the `main` to use the new constructor as follows:

```
int main()
{
 Fraction f1;
 f1.print();
 Fraction f2(3,4);
 f2.print();
 return 0;
}
```

- When one class contains 2 or more function with the same name it is referred to as **Function Overloading**
- Since we have two constructors in our example, we have **overloaded constructor**

Function to add two fractions

- Add the following function to the class so that 2 fractions can be added

```
Fraction add(const Fraction& other)
{
    Fraction result;
    result.numerator = numerator * other.denominator + other.numerator * denominator;
    result.denominator = denominator  * other.denominator;
    return result;
}
```

- To use the new function modify `main` as follows:

```
int main()
{
 Fraction f1;
 f1.print();
 Fraction f2(3,4);
 f2.print();
 cout << f1.print() << "+" << f2.print() << " = " ;
 f1.add(f2);
 f1.print();
 return 0;
}
```

Task3:

- Add 3 functions **subtract**, **multiply** and **divide** to the class
- call the newly added functions in the `main`.
- Use your favorite GPT to find the purpose of `const` and `&` in the code: `Fraction add(const Fraction& other)`