Tutorial 28 - More on C++ Friend Functions

1. Friend Function Example 1

Code Snippet:

```
class Y;
class X {
    int data;
public:
    void setValue(int value) {
        data = value;
    }
    friend void add(X, Y); // Friend function declaration
};
class Y {
    int num;
public:
    void setValue(int value) {
        num = value;
    }
    friend void add(X, Y); // Friend function declaration
};
void add(X o1, Y o2) {
    cout << "Summing data of X and Y objects gives me " << o1.data + o2.num;
}
int main() {
    X a1;
    a1.setValue(3);
    Y b1;
    b1.setValue(15);
    add(a1, b1); // Friend function call
    return 0;
}

```

Explanation:

1. **Forward Declaration**: Class `Y` is declared before defining class `X` to inform the compiler.
2. **Class X**:
   - Contains private member `data`.
   - Public method `setValue()` assigns value to `data`.
   - Declares `add()` as a friend function.
3. **Class Y**:
   - Contains private member `num`.
   - Public method `setValue()` assigns value to `num`.
   - Declares `add()` as a friend function.
4. **Friend Function** `add()`:
   - Accesses private members `data` (from `X`) and `num` (from `Y`).
   - Prints their sum.

Output:

```
Summing data of X and Y objects gives me 18
```

```
2
```

---

2. Friend Function Example 2

Code Snippet:

```
class c2;
class c1 {
    int val1;
    friend void exchange(c1 &, c2 &); // Friend function declaration
public:
    void indata(int a) {
        val1 = a;
    }
    void display() {
        cout << val1 << endl;
    }
};
class c2 {
    int val2;
    friend void exchange(c1 &, c2 &); // Friend function declaration
public:
    void indata(int a) {
        val2 = a;
    }
    void display() {
        cout << val2 << endl;
    }
};
void exchange(c1 &x, c2 &y) {
    int tmp = x.val1;
    x.val1 = y.val2;
    y.val2 = tmp;
}
int main() {
    c1 oc1;
    c2 oc2;
    oc1.indata(34);
    oc2.indata(67);
    exchange(oc1, oc2); // Friend function call
    cout << "The value of c1 after exchanging becomes: ";
    oc1.display();
    cout << "The value of c2 after exchanging becomes: ";
    oc2.display();
    return 0;
}

```

Explanation:

1. **Forward Declaration**: Class `c2` is declared before defining class `c1`.
2. **Class c1**:
   - Contains private member `val1`.
   - Public methods:
     - `indata()`: Assigns value to `val1`.
     - `display()`: Prints the value of `val1`.

- Declares `exchange()` as a friend function.

3. **Class c2**:
   - Contains private member `val2`.
   - Public methods:
     - `indata()`: Assigns value to `val2`.
     - `display()`: Prints the value of `val2`.
   - Declares `exchange()` as a friend function.
4. **Friend Function** `exchange()`:
   - Swaps values of `val1` (from `c1`) and `val2` (from `c2`).

Output:

```
The value of c1 after exchanging becomes: 67
The value of c2 after exchanging becomes: 34

```

Key Points for Notes

Friend Functions Overview

- **Definition**: Functions that can access private/protected members of a class.
- Declared with the `friend` keyword inside the class.
- Not part of the class but has access to its members.

Friend Function Example 1

1. Friend function `add()` accesses private members of two different classes (`X` and `Y`).
2. Forward declaration of class `Y` is required for defining class `X`.

Friend Function Example 2

1. Friend function `exchange()` swaps values between objects of two classes (`c1` and `c2`).
2. Forward declaration of class `c2` is required for defining class `c1`.

Usage Notes

- Use friend functions for tight coupling between classes when private data needs to be accessed directly.
- Avoid overuse to maintain encapsulation principles.