【C++】Day22

Class	C++
Date	@December 12, 2021
Material	
Series Number	
Summary	Define Nonmember Functions

【Ch8】Classes

7.1.3 Defining Nonmember Class-Related Functions

We define nonmember functions as we would any other function. As with any other function, we normally separate the declaration of the function from its definition.

Functions that are conceptually part of class, but not defined inside the class, are typically declared (but not defined) in the same header as the class itself. That way users need to include only one file to use any part of the interface.

Note: Ordinarily, nonmember functions that are part of the interface of a class should be declared in the same header as the class itself.

Defining the read and print Functions

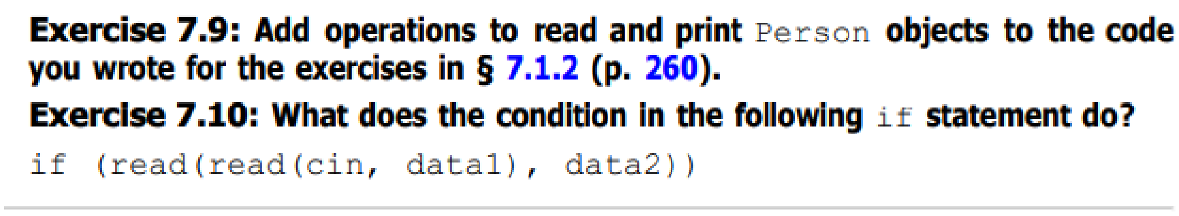
```
// input transactions contain ISBN, number of copies sold, and sales price
istream &read(istream &is, Sales_data &item) {
  double price = 0;
  is >> item.bookNo >> item.units_sold >> price;
  item.revenue = price * item.units_sold;
  return is;
}

ostream &print(ostream &os, const Sales_data &item) {
  os << item.isbn() << " " << item.units_sold << " " << item.revvenue << " " << item.avg_price();
  return os;
}
```

The `read` function reads data from the given stream into the given object. The `print` function prints the contents of the given object on the given stream.

There are two points worth nothing about these functions:

1. First, both read and write take a reference to their respective IO class types. The IO classes are types that cannot be copied, so we may only pass them by reference.
2. Reading or writing to a stream changes that stream, so both functions take ordinary references, not references to const.

Exercise



Exercise 7.9: Add operations to read and print `Person` objects to the code you wrote for the exercises in § 7.1.2 (p. 260).

Exercise 7.10: What does the condition in the following `if` statement do?

```
if (read(read(cin, data1), data2))
```

```
// Don't forget to include the header.
struct Person {
  string name;
  string address;

  const string getname() const { return name; }
  string getaddress() const { return address; }
};

istream &read(istream &is, Person &person) {
  is >> person.name >> person.address;
  return is;
}

ostream &write(ostream &os, const Person &person) {
  os << person.getname() << person.getaddress();
  return os;
}

int main() {
  Person person;
  Person person2;
  if(read(read(cin, person), person2)) {
    write(cout, person2);
    cout << endl;
    write(cout, person);
  }
  return 0;
}
```