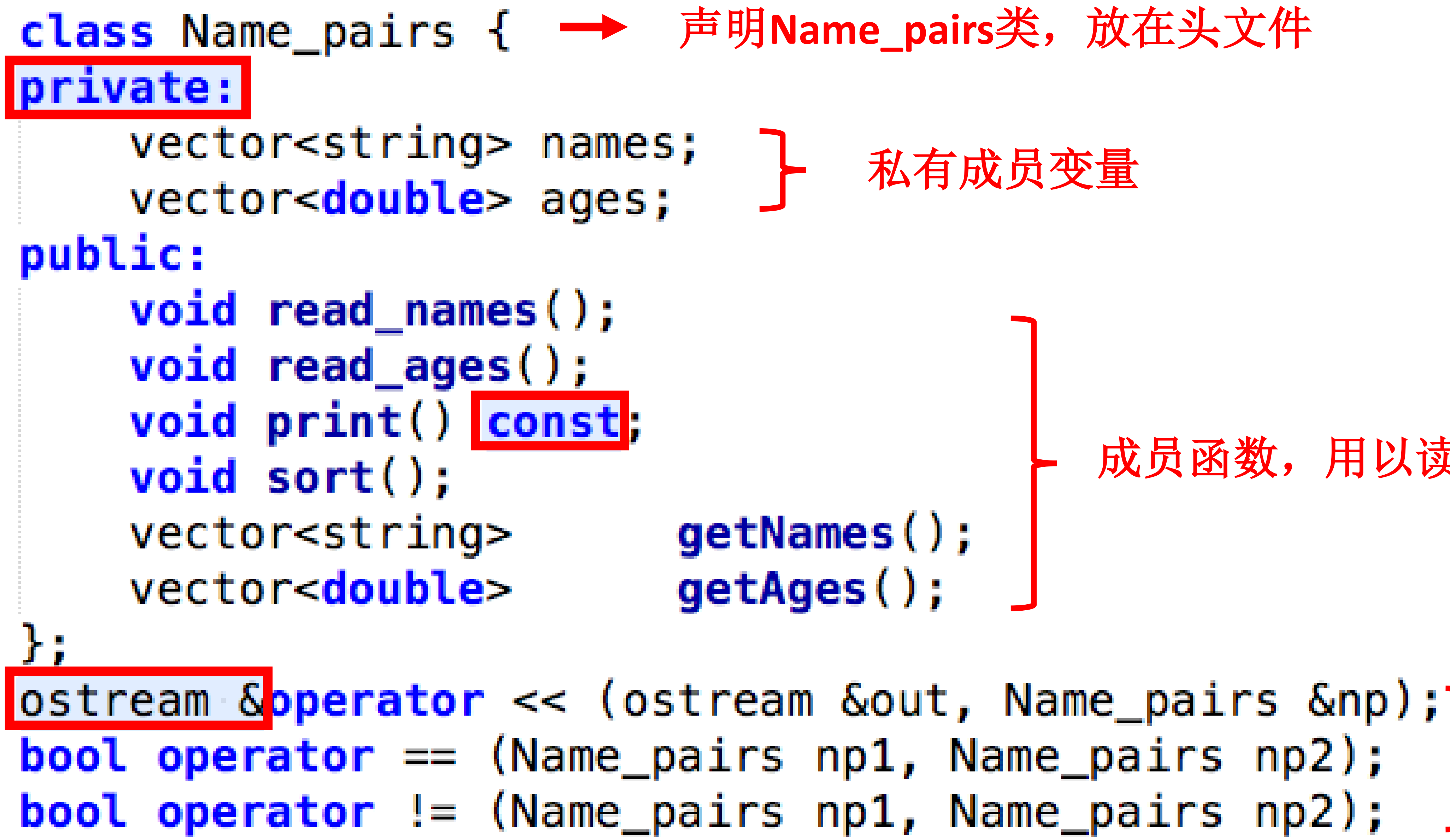
Exercise Analysis

TA-黄国快

- 1.（书中练习 P.205，2）设计并实现一个保存(name,age)对的Name_pairs类，其中name是一个string，age是一个double。
  - a)将Name_pairs类描述为包含一个名为name的vector<string>成员和一个名为age的vector<double>成员；
  - b)提供一个输入操作read_names()，能够读入一个名字列表；
  - c)提供一个输入操作read_ages()，提示用户为每个名字输入一个年龄；
  - d)提供一个print()操作，按name向量的顺序打印(name[i],age[i])对（每行一个值对）；
  - e)提供一个sort()操作将name向量按字典序排序，并相应地重排age向量，使之与name向量的新顺序相匹配。
  - 将上述所有操作全部实现为成员函数。
- 2.将Name_pairs::print()函数替换为（全局）运算符<<，并定义Name_pairs的==和!=操作符。



```
class Name_pairs {
private:
    vector<string> names;
    vector<double> ages;
public:
    void read_names();
    void read_ages();
    void print() const;
    void sort();
    vector<string>      getNames();
    vector<double>      getAges();
};
ostream &operator << (ostream &out, Name_pairs &np);
bool operator == (Name_pairs np1, Name_pairs np2);
bool operator != (Name_pairs np1, Name_pairs np2);
```

声明Name_pairs类，放在头文件

私有成员变量

成员函数，用以读写成员变量

一般函数声明

读取用户输入的名字，名字以空格作为间隔符，换行结束输入

```
//method 1: cin.get() & cin.unget()
void Name_pairs::read_names() {
    string temp;
    while (cin.get() != '\n') {
        cin.unget();
        cin >> temp;
        name.push_back(temp);
    }
}

//method 2: getline() & stringstream(header:<sstream>)
void Name_pairs::read_names() {
    string line, temp;
    getline(cin, line);
    stringstream ss(line);
    while (ss>>temp)
        name.push_back(temp);
}
```

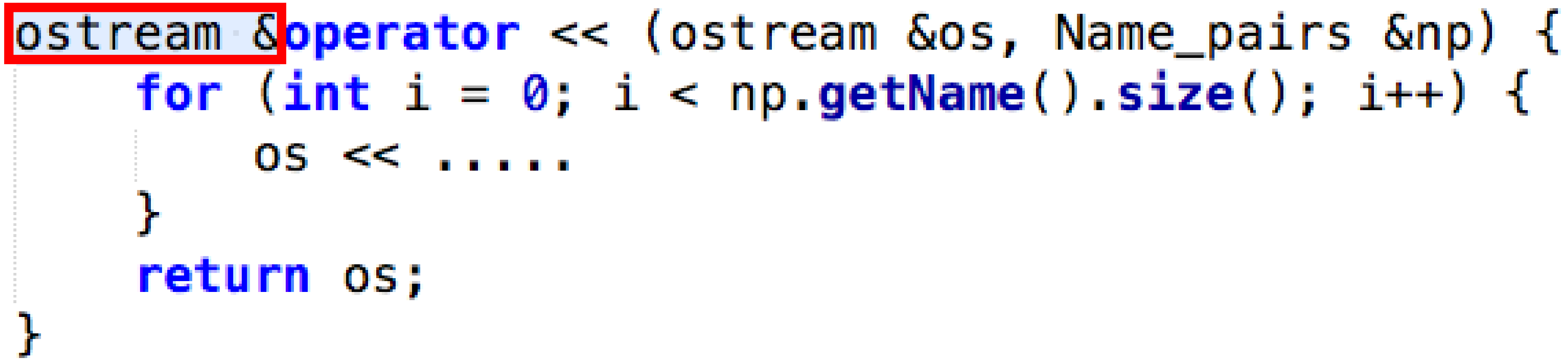
```cpp
void Name_pairs::read_ages() {
    double temp;
    for (int i = 0; i < name.size(); i++) {
        cout << "Input the age of " << name[i] << ": ";
        double temp;
        cin >> temp;
        age.push_back(temp);
    }
}

void Name_pairs::print() const {
    for (int i = 0; i < name.size(); i++) {
        cout << "(" << name[i] << "," << age[i] << ")" << endl;
    }
}

//naive sort
void Name_pair::sort() {
    int n = name.size();
    for (int i = 0; i < n; ++i)
        for (int j= i+1; j < n; ++j)
            if (name[i] > name[j]) swap(i,j);
}
```

关于操作符的重载

- 1.变量name / age应设为private，通过get()进行读取
- 2.操作符<<的重载函数返回值应为ostream&
- 3.操作符重载函数功能单一明确，不应将多个功能混在一起



```
ostream &operator << (ostream &os, Name_pairs &np) {
    for (int i = 0; i < np.getName().size(); i++) {
        os << .....
    }
    return os;
}
```

2. 定义以下几个类，注意考虑它们的继承关系：

Shape 表示图形

TwoDimensionalShape 表示二维图形

ThreeDimensionalShape 表示三维图形

Circle 表示圆，包含私有成员半径 r

Square 表示正方形，包含私有成员边长 a

Ball 表示球，包含私有成员半径 r

Cylinder 表示圆柱体，包含私有成员半径 r, 高 h

要求每个 TwoDimensionalShape 都包含 getArea() 方法，每个 ThreeDimensionalShape 都包含 getArea() 方法(计算表面积)和 getVolume() 方法写一个程序，用一个 vector<Shape *> 存放上述类的对 象。若类的实例是二维图形，则打印出它的面积；若是三维图形，则打印出它 的表面积和体积。

```cpp
class Shape {
public:
    virtual double getArea() const = 0;
};
```

$\longrightarrow$ getArea()是2D与3D图形所共有的，放在Shape里

```cpp
class TwoDimensionalShape : public Shape{};

class ThreeDimensionalShape : public Shape {
public:
    virtual double getVolume() const = 0;
};
```

$\longrightarrow$ 为3D图形声明getVolume()函数

```cpp
class Circle : public TwoDimensionalShape {
private:
    double r;
public:
    Circle(double radius);
    double getArea() const;
};

class Ball : public ThreeDimensionalShape {
private:
    double r;
public:
    Ball(double radius);
    double getArea() const;
    double getVolume() const;
};
```

子类的声明

```
int main() {
    Circle circle(1);
    Square square(1);
    Ball ball(1);
    Cylinder cylinder(1, 1);

    vector<Shape *> shapes;
    shapes.push_back(&circle);
    shapes.push_back(&square);
    shapes.push_back(&ball);
    shapes.push_back(&cylinder);

    //iterator & dynamic_cast(header:<typeinfo>)
    vector<Shape *>::iterator iShape;
    for (iShape = shapes.begin(); iShape != shapes.end(); ++i) {
        Shape *shape = *iShape;
        cout << typeid(*shape).name() << endl;
        cout << "Area:" << shape->getArea() << endl;

        ThreeDimensionalShape *tdshape =
            dynamic_cast<ThreeDimensionalShape *>(shape);
        if (tdshape)
            cout << "Volume:" << tdshape->getVolume() << endl;
    }

    return 0;
}
```

→ 使用dynamic_cast转换类型

```cpp
ostream &operator<<(ostream &os, const Shape &shape) {
    double area = shape.getArea();
    double volume = shape.getVolume();
    if (area >= 0) os << "Area:" << area << endl;
    if (volume >= 0) os << "volume:" << volume << endl;
    return os;
}
```

→ 重载<<操作符

```cpp
int main() {
    Circle circle(1);
    Square square(1);
    Ball ball(1);
    Cylinder cylinder(1, 1);

    vector<Shape*> shapes;
    shapes.push_back(&circle);
    shapes.push_back(&square);
    shapes.push_back(&ball);
    shapes.push_back(&cylinder);

    int n = shapes.size();
    for (int i = 0; i < n; ++i) {
        cout << *shapes[i];
    }

    return 0;
}
```

→ 用户无需关心Shape *指向的具体对象