SRS Setup
Login: student.turningtechnologies.com
Session ID: 20220207<A|D>

Replace <A|D> with this section's letter

Object Oriented Programming Basics II

CS 2124: Object Oriented Programming
Darryl Reeves, Ph.D.

Agenda

- OOP best practices
- The << operator
- Nested types
- In-class problem

OOP best practices

Defining a default constructor

```
class Cat {
public:
    Cat(const string& the_name, const string& the_color, double the_weight)
        : name(the_name), weight(the_weight), color(the_color) {}

    void display() const {
        cout << "Displaying a Cat named" << name << " with color ";
        cout << color << " and weight " << weight << endl;
    }

private:
    string name;
    string color;
    double weight;
};

void fill_cat_vector(ifstream& in_fs, vector<Cat>& in_vec) {
     Cat kitty;  // compilation error!

     while (in_fs >> kitty.name >> kitty.color >> kitty.weight) { // compilation error!
        in_vec.push_back(kitty);
     }
}
```

add a default constructor

Defining a default constructor

```
class Cat {
public:
    Cat(const string& the_name, const string& the_color, double the_weight)
        : name(the_name), weight(the_weight), color(the_color) {}

    Cat() {}

    void display() const {
        cout << "Displaying a Cat named" << name << " with color ";
        cout << color << " and weight " << weight << endl;
    }
private:
    string name;
    string color;
    double weight;
};
void fill_cat_vector(ifstream& in_fs, vector<Cat>& in_vec) {
     Cat kitty; // compilation error!

     while (in_fs >> kitty.name >> kitty.color >> kitty.weight) { // compilation error!
        in_vec.push_back(kitty);
     }
}
```

Defining a default constructor

```
class Cat {
public:
    Cat(const string& the_name, const string& the_color, double the_weight)
        : name(the_name), weight(the_weight), color(the_color) {}

    Cat() {}

    void display() const {
        cout << "Displaying a Cat named" << name << " with color ";
        cout << color << " and weight " << weight << endl;
    }
private:
    string name;
    string color;
    double weight;
};
void fill_cat_vector(ifstream& in_fs, vector<Cat>& in_vec) {
     Cat kitty; ~~// compilation error!~~

     while (in_fs >> kitty.name >> kitty.color >> kitty.weight) { // compilation error!
        in_vec.push_back(kitty);
     }
}
```

this one would remain

Defining a default constructor

- including a default constructor should be conscious choice
  - Should it be possible to create a Cat or Vorlon without a name?
    - Yes: default constructor may be appropriate
    - No: require that a name is provided in constructor
- constructor needs to be defined for proper initialization

"proper" has different
meanings for different classes

Class design decisions

```
Vorlon temp("");
```

empty string for `name` is no better than not providing a `name`

```
ifstream ifs("vorlons.txt");
vector<Vorlon> vor_vec;
temp.fill_vector(vor_vec, ifs);
```

`fill_vector()` should not be a Vorlon method

poor Vorlon class design

Class design decisions

```
~~Vorlon temp("");~~
```

empty string for `name` is no better than not providing a `name`

```
ifstream ifs("vorlons.txt");
vector<Vorlon> vor_vec;
~~temp.~~fill_vector(vor_vec, ifs);
```

`fill_vector()` should not be a Vorlon method

poor Vorlon class design

Class design decisions

no need for temp Vorlon

```
ifstream ifs("vorlons.txt");
vector<Vorlon> vor_vec;


fill_vector(vor_vec, ifs);
```

stand-alone function

Class design decisions

```
Vorlon temp(""), kosh("Kosh"), koshina("Koshina");
```

may want to allow Vorlons to marry

```
temp.marry(kosh, koshina);
```

role/necessity of `Vorlon temp` in marriage unclear

Class design decisions

`Vorlon` ~~`temp("")`~~`, kosh("Kosh"), koshina("Koshina");`

~~`temp.`~~`marry(kosh, koshina);`

role/necessity of `Vorlon`
`temp` in marriage unclear

may want to allow
Vorlons to marry

Class design decisions

```
Vorlon kosh("Kosh"), koshina("Koshina");

kosh.marry(koshina);
```

simplified design with clear association between objects

The << operator

Printing structs

```
struct Cat {
   string color = "brown;
   string name = "Charlie";
   double weight = 6.5;
};
```

```
Cat my_cat;

cout << my_cat << endl;
```

will learn implementation later

won't work! (results in compilation error)

```
void print_cat(const Cat& kitty) {
   cout << kitty.name << ' ' << kitty.color << ' ' << kitty.weight << endl;
}
```

Object output

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   void display() const {
       cout << "Displaying a Cat named" << name << " with color ";
       cout << color << " and weight " << weight << endl;
   }
private:
    string name;
    string color;
    double weight;
};
```

Object output

```
Cat my_cat("Whiskers", "brown", 8);

my_cat.display()
```

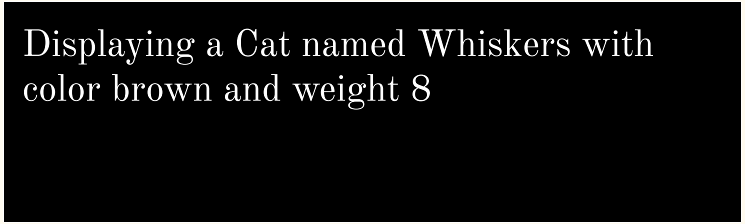
```
Displaying a Cat named Whiskers with
color brown and weight 8
```

Object output

```
Cat my_cat("Whiskers", "brown", 8);
```

~~`my_cat.display()`~~

```
cout << my_cat << endl;
```



```
Displaying a Cat named Whiskers with
color brown and weight 8
```

Want this behavior

The output operator `<<`

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   void display() const {
       cout << "Displaying a Cat named" << name << " with color ";
       cout << color << " and weight " << weight << endl;
   }

private:
    string name;
    string color;
    double weight;

};
```

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   ~~void display() const {~~
   ~~    cout << "Displaying a Cat named" << name << " with color ";~~
   ~~    cout << color << " and weight " << weight << endl;~~
   ~~}~~

private:
    string name;
    string color;
    double weight;
};

ostream& operator<< () { }
```

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   ~~void display() const {~~
   ~~    cout << "Displaying a Cat named" << name << " with color ";~~
   ~~    cout << color << " and weight " << weight << endl;~~
   ~~}~~

private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os) { }
```

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   ~~void display() const {~~
   ~~    cout << "Displaying a Cat named" << name << " with color ";~~
   ~~    cout << color << " and weight " << weight << endl;~~
   ~~}~~
private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) { }
```

The output operator `<<`

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   ~~void display() const {~~
~~       cout << "Displaying a Cat named" << name << " with color ";~~
~~       cout << color << " and weight " << weight << endl;~~
   ~~}~~

private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    // output Cat rhs to ostream
}
```

The output operator `<<`

return type is `ostream&` (ostream reference)

first parameter is `ostream&` (ostream reference)

second parameter is `const Cat&`

```
ostream& operator<< (ostream& os, const Cat& rhs) {

    // output Cat rhs to ostream
    // return modified ostream
}
```

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   ~~void display() const {~~
   ~~    cout << "Displaying a Cat named" << name << " with color ";~~
   ~~    cout << color << " and weight " << weight << endl;~~
   ~~}~~

private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.name << " with color ";
    os << rhs.color << " and weight " << rhs.weight << endl;
    return os;
}
```

compilation error!

TurningPoint

SRS Setup
Login: student.turningtechnologies.com
Session ID: 20220207<A|D>

Replace <A|D> with this section's letter

Why does the definition of `operator<<` result in a compilation error?

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}
private:
    string name;
    string color;
    double weight;
};
```

```
ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.name << " with color ";
    os << rhs.color << " and weight " << rhs.weight << endl;

    return os;
}
```

compilation error!

The output operator `<<`

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}
private:
    string name;
    string color;
    double weight;
};
```

add public methods for accessing private members

```
ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.name << " with color ";
    os << rhs.color << " and weight " << rhs.weight << endl;

    return os;
}
```

compilation error!

The output operator `<<`

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}
   const string& get_name() const { return name; }
   const string& get_color() const { return color; }
   double get_weight() const { return weight; }
private:
    string name;
    string color;
    double weight;
};
```

accessor methods

const & avoids making copy of string

```
ostream& operator<< (ostream& os, const Cat& rhs) {
     os << "Displaying a Cat named" << rhs.name << " with color ";
     os << rhs.color << " and weight " << rhs.weight << endl;

     return os;
}
```

compilation error!

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}
   const string& get_name() const { return name; }
   const string& get_color() const { return color; }
   double get_weight() const { return weight; }
private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.get_name() << " with color ";
    os << rhs.color << " and weight " << rhs.weight << endl;

    return os;
}
```

compilation error!

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   const string& get_name() const { return name; }

   const string& get_color() const { return color; }

   double get_weight() const { return weight; }

private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.get_name() << " with color ";
    os << rhs.get_color() << " and weight " << rhs.weight << endl;

    return os;
}
```

compilation error!

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   const string& get_name() const { return name; }

   const string& get_color() const { return color; }

   double get_weight() const { return weight; }

private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.get_name() << " with color ";
    os << rhs.get_color() << " and weight " << rhs.get_weight() << endl;

    return os;
}
```

compilation error!

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}
   const string& get_name() const { return name; }
   const string& get_color() const { return color; }
   double get_weight() const { return weight; }
private:
    string name;
    string color;
    double weight;
};
ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.get_name() << " with color ";
    os << rhs.get_color() << " and weight " << rhs.get_weight() << endl;

    return os;
}
```

~~compilation error!~~

The output operator <<

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}
   const string& get_name() const { return name; }
   const string& get_color() const { return color; }
   double get_weight() const { return weight; }
private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.get_name() << " with color ";
    os << rhs.get_color() << " and weight " << rhs.get_weight() << endl;

    return os;
}
```

Object output

```
Cat my_cat("Whiskers", "brown", 8);
```

~~my_cat.display()~~

```
cout << my_cat << endl;
```

Object output

```
Cat my_cat("Whiskers", "brown", 8);

cout << my_cat << endl;
```

```
Displaying a Cat named Whiskers with
color brown and weight 8
```

Friend status

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   const string& get_name() const { return name; }

   const string& get_color() const { return color; }

   double get_weight() const { return weight; }

private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.get_name() << " with color ";
    os << rhs.get_color() << " and weight " << rhs.get_weight() << endl;

    return os;
}
```

accessor methods

alternative implementation available

Friend status

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}

   ~~const string& get_name() const { return name; }~~
   ~~const string& get_color() const { return color; }~~
   ~~double get_weight() const { return weight; }~~
private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
     os << "Displaying a Cat named" << rhs.~~get_name()~~ << " with color ";
     os << rhs.~~get_color()~~ << " and weight " << rhs.~~get_weight()~~ << endl;

     return os;
}
```

alternative implementation available

Friend status

```
class Cat {
public:
   Cat(const string& the_name, const string& the_color, double the_weight)
       : name(the_name), weight(the_weight), color(the_color) {}




private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.___ << " with color ";
    os << rhs.___ << " and weight " << rhs.___ << endl;

    return os;
}
```

Friend status

```
class Cat {

    // give operator<< function access to Cat private member data

public:
    Cat(const string& the_name, const string& the_color, double the_weight)
        : name(the_name), weight(the_weight), color(the_color) {}


private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.___ << " with color ";
    os << rhs.___ << " and weight " << rhs.___ << endl;

    return os;
}
```

Friend status

```
class Cat {

    ostream& operator<<(ostream&, const Cat&);
```

function prototype

```
public:
    Cat(const string& the_name, const string& the_color, double the_weight)
        : name(the_name), weight(the_weight), color(the_color) {}


private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.___ << " with color ";
    os << rhs.___ << " and weight " << rhs.___ << endl;

    return os;
}
```

Friend status

gives function access to Cat's private member variables

```
class Cat {

    friend ostream& operator<<(ostream&, const Cat&);

public:
    Cat(const string& the_name, const string& the_color, double the_weight)
        : name(the_name), weight(the_weight), color(the_color) {}


private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.___ << " with color ";
    os << rhs.___ << " and weight " << rhs.___ << endl;

    return os;
}
```

Friend status

gives function access to Cat's private member variables

function modifier -- not a return type

```
class Cat {

    friend ostream& operator<<(ostream&, const Cat&);

public:
    Cat(const string& the_name, const string& the_color, double the_weight)
        : name(the_name), weight(the_weight), color(the_color) {}


private:
    string name;
    string color;
    double weight;
};

ostream& operator<< (ostream& os, const Cat& rhs) {
    os << "Displaying a Cat named" << rhs.name << " with color ";
    os << rhs.color << " and weight " << rhs.weight << endl;

    return os;
}
```

Nested types

Including objects in a class

```
class Vorlon {
public:
    Vorlon(const string& a_name) : my_name(a_name) {}
    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
};
```

instance of
string `class`

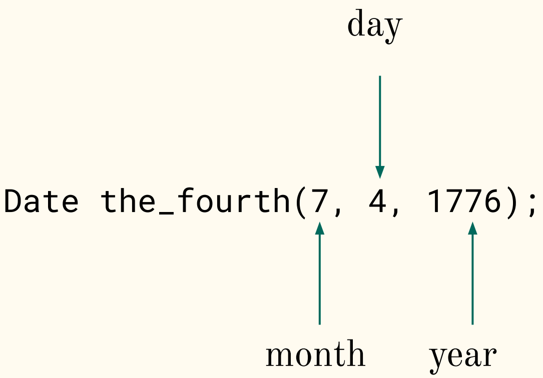
string objects available
with #include <string>

Including objects in a class

```
class Vorlon {
public:
    Vorlon(const string& a_name) : my_name(a_name) {}
    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    // birth date member variable
};
```

an instance of a
user-defined class

A `Date` class



day

```
Date the_fourth(7, 4, 1776);
```

month

year

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};
```

Including objects in a class

```
class Vorlon {
public:
    Vorlon(const string& a_name) : my_name(a_name) {}
    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    // birth date member variable
};
```

an instance of a
user-defined class

Including objects in a class

```
class Vorlon {
public:
    Vorlon(const string& a_name) : my_name(a_name) {}
    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    Date bday;
};
```

Including objects in a class

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}

    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};

class Vorlon {
public:
    Vorlon(const string& a_name) : my_name(a_name) {}
    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    Date bday;
};
```

Two options for initializing `bday`

1. pass a `Date` object to `Vorlon` constructor
2. pass `Date` constructor values to `Vorlon` constructor

Including objects in a class

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}

    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};

class Vorlon {
public:
    Vorlon(const string& a_name) : my_name(a_name) {}
    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    Date bday;
};
```

Two options for initializing `bday`

1. ~~pass a `Date` object to `Vorlon` constructor~~
2. pass `Date` constructor values to `Vorlon` constructor

Including objects in a class

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}

    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};

class Vorlon {
public:
    Vorlon(const string& a_name, int b_month, int b_day, int b_year) : my_name(a_name) {}
    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    Date bday;
};
```

Two options for initializing `bday`

1. ~~pass a `Date` object to `Vorlon` constructor~~
2. pass `Date` constructor values to `Vorlon` constructor

Including objects in a class

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}

    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};

class Vorlon {
public:
    Vorlon(const string& a_name, int b_month, int b_day, int b_year)
    : my_name(a_name), bday(b_month, b_day, b_year) {}

    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    Date bday;
};
```

Two options for initializing `bday`

1. ~~pass a `Date` object to `Vorlon` constructor~~
2. pass `Date` constructor values to `Vorlon` constructor

```
Vorlon kosh("Kosh", 3, 14, 1592);
```

Defining a nested class

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}

    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};
```

alternatively Date class can
be defined within Vorlon class

```
class Vorlon {
public:
    Vorlon(const string& a_name, int b_month, int b_day, int b_year)
    : my_name(a_name), bday(b_month, b_day, b_year) {}

    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    Date bday;
};
```

Defining a nested class

```
class Vorlon {

    class Date {
    public:
        Date(int month, int day, int year)
            : month(month), day(day), year(year) {}

        void display() const {
            cout << month << '/' << day << '/' << year;
        }
    private:
        int month, day, year;
    };

public:
    Vorlon(const string& a_name, int b_month, int b_day, int b_year)
    : my_name(a_name), bday(b_month, b_day, b_year) {}

    void display() {
        cout << "Displaying a Vorlon named " << my_name << endl;
    }
private:
    const string my_name;
    Date bday;
};
```

Date is private class of Vorlon

`Date` is only accessible through `Vorlon` class

- must use `Vorlon::Date`

scope resolution operator

In-class problem

Outputting objects

person.cpp

```cpp
class Person {
public:
    Person(const string& the_name) : name(the_name) {}

    void eat() const { cout << name << " eating\n"; }

    void set_name(const string& the_name) { name = the_name; }

private:
    string name;
};


int main() {
    Person john("John");

    john.eat();
}
```

```
% g++ -std=c++11 person.cpp -o person
% ./person
John eating
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
};


int main() {
    Person john("John");

    john.eat();
}
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    // member variable to track birthday
};


int main() {
    Person john("John");

    john.eat();
}
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    // member variable to track birthday
};


int main() {
    Person john("John");

    // output John's name and birthday
}
```

Outputting objects

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}

    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    // member variable to track birthday
};


int main() {
    Person john("John");

    // output John's name and birthday
}
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};


int main() {
    Person john("John");

    // output John's name and birthday
}
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};


int main() {
    Person john("John");

    cout << john << endl;
}
```

desired behavior

```
% g++ -std=c++11 person.cpp -o person
% ./person
Person: name = John, dob = 7/14/1920
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};



int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Why does the attempt to output the `Person` object, `john`, result in a compilation error?

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};



int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Person {
public:
    Person(const string& the_name) : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};


int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};



int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};



int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

Outputting objects

```
class Person {
    // output details of Person object

public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ ___
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ _1_
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Which function name replaces blank #1 to output a `Person` object?

```
class Person {
    // output details of Person object
    ___ ___ _1_
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<() {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(___ ___, ___ ___) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(___ os, ___ rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(_2_ os, ___ rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

What is the type of the first parameter of the `operator<<` function (replacing blank #2)?

```
class Person {
    // output details of Person object
    ___ ___ operator<<(_2_ os, ___ rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, ___ rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, _3_ rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

What is the type of the second parameter of the `operator<<` function (replacing blank #3)?

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, _3_ rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, ___ Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, _4_ Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

Which keyword replaces blank #4 to ensure the second parameter of the `operator<<` function cannot be modified?

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, _4_ Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ___ operator<<(ostream& os, const Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ _5_ operator<<(ostream& os, const Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;  compilation error
}
```

What is the return type of the `operator<<` function (replacing blank #5)?

```
class Person {
    // output details of Person object
    ___ _5_ operator<<(ostream& os, const Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Person {
    // output details of Person object
    ___ ostream& operator<<(ostream& os, const Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    // output details of Person object
    _6_ ostream& operator<<(ostream& os, const Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Which modifier needs to be added to the `operator<<` function to access private members of a `Person` object?

```
class Person {
    // output details of Person object
    _6_ ostream& operator<<(ostream& os, const Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Person {
    // output details of Person object
    friend ostream& operator<<(ostream& os, const Person& rhs) {}
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};

int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {

    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
int main() {
    Person john("John");

    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        ___
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
int main() {
    Person john("John");
    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return _7_;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
int main() {
    Person john("John");
    cout << john << endl;
}
```

compilation error

Which object should be returned by `operator<<` (replacing blank #7)?

```
class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return _7_;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
int main() {
    Person john("John");
    cout << john << endl;
}
```

compilation error

Outputting objects

```
class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return os;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
int main() {
    Person john("John");
    cout << john << endl;  ~~compilation error~~
}
```

Outputting objects

```
class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return os;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
int main() {
    Person john("John");
    cout << john << endl;
}
```

compilation error

What causes the compilation error from the implementation of `Person` class's `operator<<` function?

```
class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return os;
    }

    ...
};

class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};
```

compilation error

Outputting objects

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
    void display() const {
        cout << month << '/' << day << '/' << year;
    }
private:
    int month, day, year;
};
```

Outputting objects

```
class Date {
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
    ~~void display() const {~~
~~        cout << month << '/' << day << '/' << year;~~
~~    }~~
private:
    int month, day, year;
};
```

Outputting objects

```
class Date {
    // implement operator<< function
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const ___ rhs) {
        os << rhs.month << '/' << rhs.day << '/' << rhs.year;
        return os;
    }

public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const ___ rhs) {
        os << rhs.month << '/' << rhs.day << '/' << rhs.year;
        return os;
    }

public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const _8_ rhs) {
        os << rhs.month << '/' << rhs.day << '/' << rhs.year;
        return os;
    }

public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};
```

Which type replaces blank #8 to complete the definition of `operator<<` in the Date class?

```
class Date {
    friend ostream& operator<<(ostream& os, const _8_ rhs) {
        os << rhs.month << '/' << rhs.day << '/' << rhs.year;
        return os;
    }

public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const Date& rhs) {
        os << rhs.month << '/' << rhs.day << '/' << rhs.year;
        return os;
    }

public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const Date& rhs) {
          os << rhs.month << '/' << rhs.day << '/' << rhs.year;
          return os;
    }
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};

class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return os;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
```

~~compilation error~~

```
int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const Date& rhs) {
          os << rhs.month << '/' << rhs.day << '/' << rhs.year;
          return os;
    }
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};

class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return os;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
```

```
int main() {
    Person john("John");

    cout << john << endl;
}
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const Date& rhs) {
          os << rhs.month << '/' << rhs.day << '/' << rhs.year;
          return os;
    }
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};

class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return os;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
```

need to provide birth date

```
int main() {
    Person john("John" );

    cout << john << endl;
}
```

Outputting objects

```
class Date {
    friend ostream& operator<<(ostream& os, const Date& rhs) {
          os << rhs.month << '/' << rhs.day << '/' << rhs.year;
          return os;
    }
public:
    Date(int month, int day, int year)
        : month(month), day(day), year(year) {}
private:
    int month, day, year;
};

class Person {
    friend ostream& operator<<(ostream& os, const Person& rhs) {
        os << "Person: name = " << rhs.name << ", dob = " << rhs.dob;
        return os;
    }
public:
    Person(const string& the_name, int b_month, int b_day, int b_year)
        : name(the_name), dob(b_month, b_day, b_year) {}
    void eat() const { cout << name << " eating\n"; }
    void set_name(const string& the_name) { name = the_name; }
private:
    string name;
    Date dob;
};
```

```
int main() {
    Person john("John", 7, 14, 1920);

    cout << john << endl;
}
```

```
% g++ -std=c++11 person.cpp -o person
% ./person
Person: name = John, dob = 7/14/1920
```