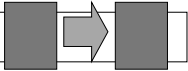


Copy Constructor (CC)



• **Special constructor**

- Can accept single argument
 - Reference (almost always const) to an object of the same class

• **Used automatically**

- When compiler must copy an object
- For call by value or return by value

Copy Constructor Example

```
Rational::Rational
    (const Rational& right)
{
    num = right.num;
    den = right.den;
    cout << "Copy constructor ";
}
void f(Rational k);
    Rational r (5,3);
    f(r);
void f(Rational k)
{
    cout << k;
}
```

Initialize data members from "right"

Assignment Operator

• **Permits objects to be assigned**

- E.g., "a = b"

• **Must be a member function**

• **Object itself is "left" operand**

• **Single argument is "right" operand**

- Constant reference to another object of the same type (or other type?)

Assignment Operator Definition

```

[... ]Rational::operator=
  ((const Rational& right))
{
  num = right.num;
  den = right.den;
  return ???;
}

```

Annotations:

- Right operand (points to `(const Rational& right)`)
- Assign members (points to `num = right.num;` and `den = right.den;`)
- What about return value? (points to `return ???;`)

Chained Assignment

```

Rational r;
Rational s;
Rational t (4,7);

r = s = t;
r = (s = t);

```

Annotations:

- Chained assignment (points to `r = s = t;`)
- Equivalent (right-associative) (points to `r = (s = t);`)
- Return value from operator=? (points to `(s = t)` in the second line)

Completed Assignment Operator

```

Rational&
Rational::operator=
  (const Rational& right)
{
  num = right.num;
  den = right.den;
  return (*this);
}

```

Annotations:

- Return (const?) reference to "self" (points to `return (*this);`)

C++ keyword `this` is a "pointer" to (i.e., position of) current object; `*this` is "self"

Destructor

- **Another special member function**
- **Opposite of constructor**
 - Called automatically just before object goes out of existence
- **Special name**
 - Same as class name, prefixed with tilde
 - E.g., “~Rational()”



Destructor Definition

```
Rational::~~Rational()
{
  // Nothing special to do
  // for Rational object.
}
```

More complex objects may have to clean up (e.g., disconnect network connection or release memory) before they are destroyed.

Compiler-Generated Functions

- **If no explicit:**
 - Copy constructor
 - Assignment operator
- **Compiler will generate one**
- **Memberwise “copy”**
 - Built-in types: bit copy
 - Class types: invoke member CC or assignment operator



Need Explicit CC/Assignment?

- **Compiler can generate**
 - But sometimes will do wrong thing
 - With “pointers”, dynamic memory, etc. (more on this later)
- **General policy**
 - Provide explicit “big 3” (or “big 4”)
 - CC, assignment, destructor
 - No-argument constructor

Static Data Members

- **Data object common to class**
- **Analogy: apartment building**
 - Separate bathtubs (normal member)
 - Common swimming pool (static)
- **Example**
 - Serial number for gold bars
 - Normal member: serial number
 - Static member: next serial number

Gold Bar Class Definition

```
// goldbar.h
class GoldBar
{
public:
  GoldBar();
  int GetSerial() const;
private:
  int serial;
  static int next_serial;
};
```

Gold Bar Class Implementation

```

// goldbar.cpp
int GoldBar::next_serial = 1;

GoldBar::GoldBar ()
{
  serial = next_serial++;
}

int GoldBar::GetSerial() const
{
  return serial;
}

```

Static member initialization

Horizontal lines for notes

Static Member Functions

- Ordinary member functions
 - Tied to a particular object
 - Must be called as "obj.func()"
- Static member functions
 - Belong to class as a whole
 - Don't require a particular object

Horizontal lines for notes

Static Member Function (1)

```

// goldbar.h
class GoldBar
{
public:
  GoldBar();
  int GetSerial() const;
  static int GetNextSerial();
private:
  int serial;
  static int next_serial;
};

```

Static member function declaration

Horizontal lines for notes

Static Member Function (2)

Static member function definition

// goldbar.cpp

```
int GoldBar::GetNextSerial()
{
    return next_serial;
};
```

Static Member Function Usage

// main.cpp

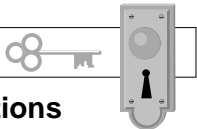
Call with or without object

```
GoldBar bar_one;
int num;

num = GoldBar::GetNextSerial();
num = bar_one.GetNextSerial();
```

Can call even if NO GoldBar objects exist!

Member Access



• Ordinary member functions

- Access all data members
- Call all member functions

• Static member functions

- Access only static data members
- Call only static member functions
