

Copy & Assignment

CMSC 202

Copying Objects...

When does C++ make copies of objects?

Pass by value

Return by value

Assignment

and...

New object initialized from existing object

Haven't seen this yet... but it is very useful

Copy Constructor

Initialize an object based on an existing object

Examples:

```
int a = 7;
```

```
int b(a); // Copy constructor
```

```
Shoe shoeOfMJ( "Nike", 16 );
```

```
Shoe myShoe( shoeOfMJ ); // Copy
```

Copy Constructor

Use when dynamic memory is allocated

Syntax:

Prototype:

```
ClassName( const ClassName& obj );
```

Implementation:

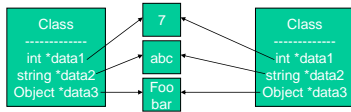
```
ClassName::ClassName( const ClassName& obj )  
{  
    // code to dynamically allocate data  
}
```

Why do we care?

Remember

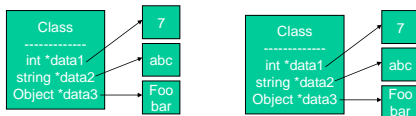
Assignment (by default) makes a direct copy of data members...

With dynamic memory – this would be copying pointers



What do we want?

Each object should have own memory allocated to members...



Example

```
class Shoe
{
public:
    Shoe( const Shoe& shoe );
private:
    int *m_size;
    string *m_brand;
};

Shoe::Shoe( const Shoe& shoe )
{
    m_size = new int( *shoe.m_size );
    m_brand = new string( *shoe.m_brand );
}
```

What's going on here?

What else?

Assignment Operator

Define if using dynamic memory

Syntax:

Prototype:

```
ClassName& operator=( const ClassName& obj );
```

Definition:

```
ClassName& ClassName::operator=( const ClassName& obj )
{
    // Deallocate existing memory, if necessary
    // Allocate new memory
}
```

What's wrong with this?

```
Shoe& Shoe::operator=(
    const Shoe& shoe )
{
    m_size =
        new int(*shoe.m_size);
    m_brand =
        new string(*shoe.m_brand);
}

// In main()
Shoe a(7, "abc");
Shoe b(4, "edf");

b = a;
```

What happened to the memory b was pointing to first???

What's wrong with this?

```
void Shoe::operator=( const Shoe& shoe )
{
    m_size = *shoe.m_size;
    m_brand = *shoe.m_brand;
}

Shoe a(7, "abc");
Shoe b(4, "edf");
Shoe c(9, "ghi");

c = b = a;
```

How does the `c = b` work, when `b = a` returns nothing??

Fixed

```
Shoe& Shoe::operator=( const Shoe& shoe )
{
    m_size = *shoe.m_size;
    m_brand = *shoe.m_brand;

    return *this;
}

Shoe a(7, "abc");
Shoe b(4, "edf");
Shoe c(9, "ghi");

c = b = a;
```

What's this?
this – a pointer to the current object

Self-Assignment

```
class RentalSystem {
public:
    // Assume constructor, other methods...
    RentalSystem& operator=(
        const RentalSystem& rs )
private:
    Customer *m_customers;
    int m_nbrOfCustomers;
};

RentalSystem& RentalSystem::operator=(
    const RentalSystem& rs )
{
    delete [] m_customers;

    m_customers = new Customer[rs.m_nbrOfCustomers];
    for (int i = 0; i < rs.m_nbrOfCustomers; ++i)
        m_customers[i] = rs.m_customers[i];

    return *this;
}
```

What happens when you do the following?

```
RentalSystem r;
// Add customers...
r = r;
```

Protect from Self-assignment

```
RentalSystem& RentalSystem::operator=(
    const RentalSystem & rs )
{
    // If this is NOT the same object as rs
    if ( this != &rs )
    {
        delete [] m_customers;

        m_customers = new Customer[rs.m_nbrOfCustomers];
        for (int i = 0; i < rs.m_nbrOfCustomers; ++i)
            m_customers[i] = rs.m_customers[i];
    }

    return *this;
}
```

Practice

Implement copy constructor and = operator

```
class Stapler
{
public:
    _____ // copy constructor
    _____ // operator=
private:
    int *m_nbrStaples;
};
```
