Announcements

MP2 available, due 9/13, 11:59p.

Parameter passing:

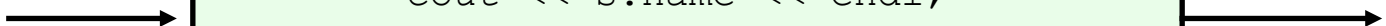
```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

What happens when we run code like this:

```
int main() {
    student a;
    print_student1(a);
}
```

?

```
bool print_student1(student s){
    if (!s.printed)
        cout << s.name << endl;
    return true;
}
```



Parameter passing:

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

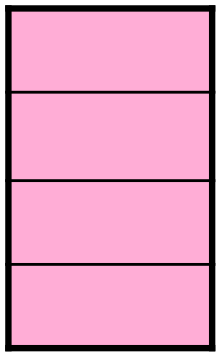
Function defn

```
void print_student3(student    s){
    if (!  s.printed)
        cout <<   s.name << endl;

}
```

Example of use

```
student c;
… // initialize c
print_student3(c);
cout << c.printed << endl;
```



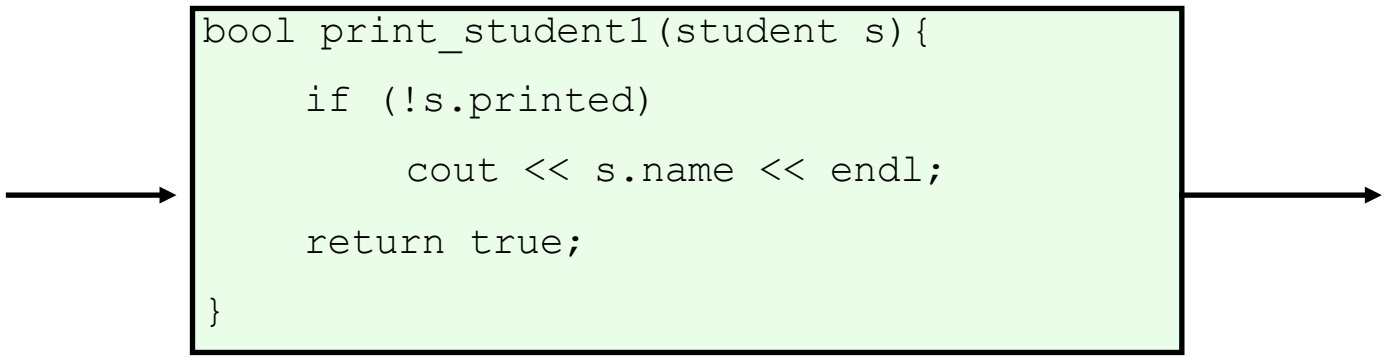
Return values:

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

What happens when we run code like this:

```
int main() {
    student a;
    bool b = print_student1(a);
}
```

?



Return by _____ or _____ or _____ .

Returns:

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

Function defn

```
student * print_student5(student s){
    student w = s;
    if (!w.printed){
       cout << w.name << endl;
       w.printed = true;
    }
    return &w;
}
```

Example of use

```
student c;
student * d;
… // initialize c
d = print_student5(c);
```

Returns:

```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

Function defn

```
student & print_student5(student s){
    student w = s;
    if (!w.printed){
       cout << w.name << endl;
       w.printed = true;
    }
    return w;
}
```

Example of use

```
student c,d;
… // initialize c
d = print_student5(c);
```

Lesson: don't return 1) a pointer to a local variable, nor 2) a local variable by reference.

Constructors reprise:

```
class sphere{

public:

sphere();

sphere(double r);

sphere(const sphere & orig);

void setRadius(double newRad);

double getDiameter() const;

…

private:

double theRadius;

int numAtts;

string * atts;

};
```

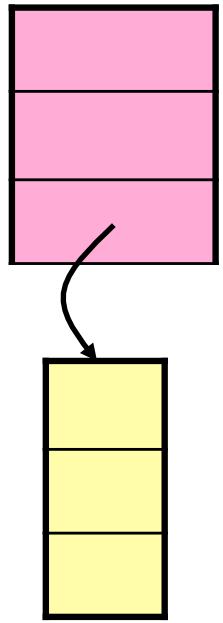
```
…
//default constructor, alt syntax
sphere::sphere()
{




}
…
```

What do you want the object to look like when you declare it?

```
sphere a;
```



Copy constructor - utility:

```
class sphere{

public:

sphere();

sphere(double r);

sphere(const sphere & orig);

void setRadius(double newRad);

double getDiameter() const;

…

private:

double theRadius;

int numAtts;

string * atts;

};
```

Use 1:

```
sphere myFun(sphere s){
   //play with s
   return s;
}

int main(){
  sphere a, b;
   // initialize a
   b = myFun(a);
   return 0;
}
```

Use 2:

```
int main(){




};
```

Copy constructor:

```
class sphere{

public:

sphere();

sphere(double r);

sphere(const sphere & orig);

void setRadius(double newRad);

double getDiameter() const;

…

private:

double theRadius;

int numAtts;

string * atts;

};
```

```
…
//copy constructor
sphere::sphere(const sphere & orig)
{



}
…
```