C++ Short Course Part 1

- Classes
- Pointers
- Arrays
- Parameter passing
- Return values

C++ Short Course Part 1

- Classes
- Pointers
- Arrays
- Parameter passing
- Return values

Classes in C++:

Every variable has _____, _____, _____, _____

Primitive types:

```
int myFavoInt;
char rating = 'E';
double u = 37.;
```

User defined types:

```
sphere myFavoSphere;
______ is a group of ________ and _________
```

Structure of a class defn:

how do we implement `sphere myFavoSphere;` ?

```
class sphere{

//member declarations

…

};
```

`sphere` member function definitions.

Structure of a class defn (cont):

```
class sphere{
public:




private:


};
```

`sphere` representation:

`sphere` functionality:

1.

2.

3.

```
int main(){



}
```

Structure of a class defn (cont):

```
class sphere{

public:

sphere();

sphere(double r);

void setRadius(double newRad);

double getDiameter() const;

…



private:

double theRadius;

};
```

```
//constructor(s) (next page)


void sphere::setRadius(double newRad){


}

double sphere::getDiameter() const {


}

...
```

Asides:

_____:

__:

Constructors (intro): When you *declare* a sphere, a sphere class constructor is invoked.

Points to remember abt ctors:

1.

2.

3.

```
int main(){



}
```

```
…
//default constructor
sphere::sphere(){

}

//default constructor, alternative
sphere::sphere()
{

}

//constructor with given radius
sphere::sphere(double r){




}
...
```

Class Definition… where are we?

Stepping back…

Ideas/concepts:

- Class definitions
- Class function implementation
- Constructors
- Clients

OOP: we now understand how C++ supports

- Inheritance
- Encapsulation (separation of interface from implementation)
  1)
  2)
- Polymorphism

Our first class…

sphere.h

```
class sphere{

};
```

What surprises you about this code?

main.cpp

```
#include "sphere.h"

int main(){
   sphere a;
}
```

1. Upon command `> clang++ main.cpp` does this code compile?
2. Upon command `> ./a.out` does it run?

Access control and encapsulation:

sphere.h

```
class sphere{

   double theRadius;

};
```

What surprises you about this code?

main.cpp

```
#include "sphere.h"
#include <iostream>
using namespace std;

int main(){
   sphere a;
   cout << a.theRadius << endl;
}
```

1. Upon command `> clang++ main.cpp` does this code compile?
2. Upon command `> ./a.out` does it run?
3. In c++ class members are, by default, "private". Why would we want to hide our representation of an object from a client?
4. How many collaborators are you allowed to have for PAs in this cpsc221?

C++ Short Course Part 1

- Classes
- Pointers
- Arrays
- Parameter passing
- Return values

Switching gears…

Configure your iMac 27-inch

Use the options below to build the system of your dreams

Memory

More memory (RAM) increases performance and enables your computer to perform faster and better. Choose additional 1066MHz DDR3 memory for your iMac.

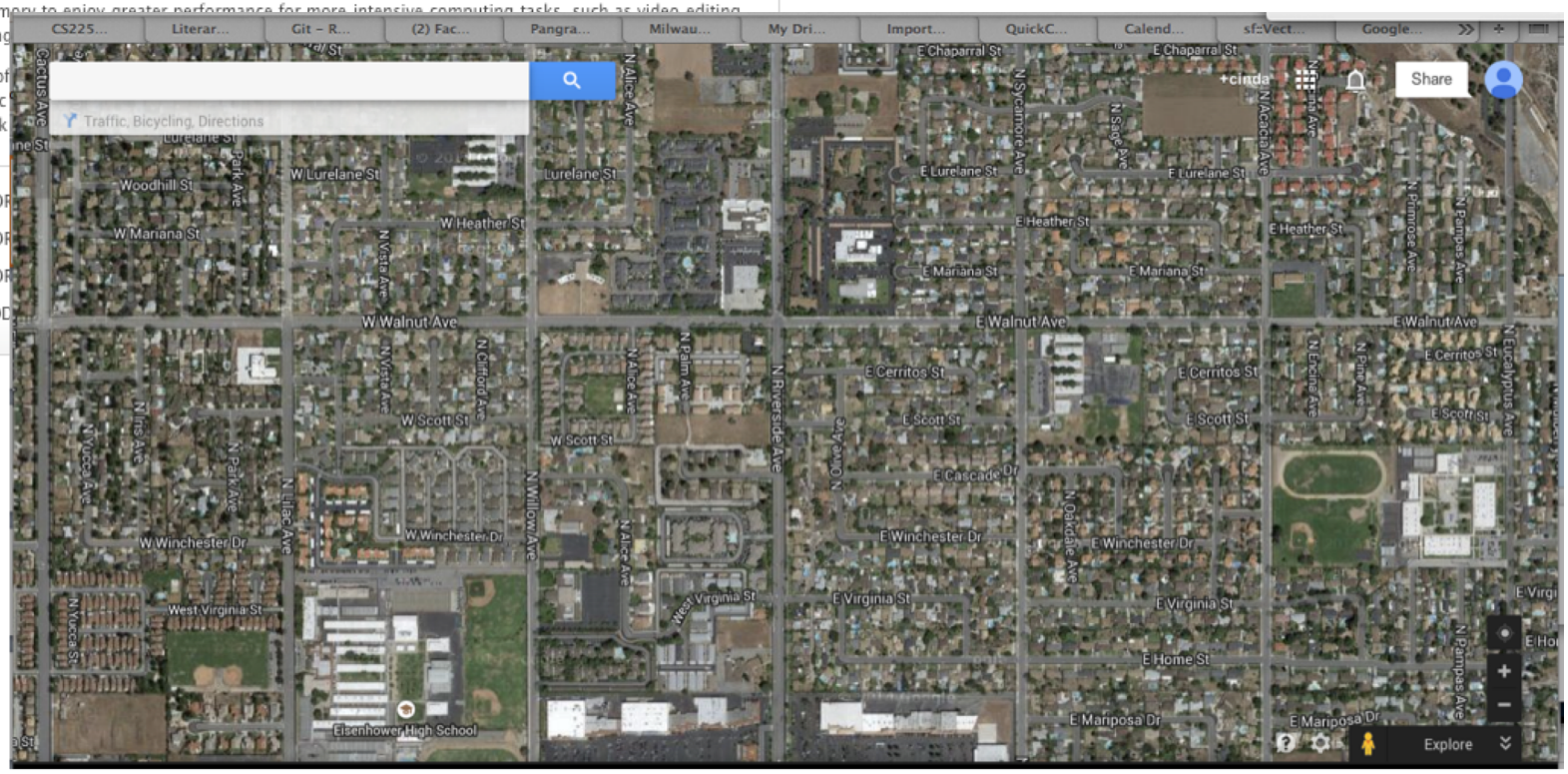
Learn more

The more memory your computer has, the more programs you can run simultaneously, and the better performance you get from your computer.

- Select the standard memory configuration to support day-to-day tasks such as email, word processing, and web browsing as well as more complex tasks such as editing photos, creating illustrations, and building presentations.
- Upgrade your memory to enjoy greater performance for more intensive computing tasks, such as video editing and DVD authoring

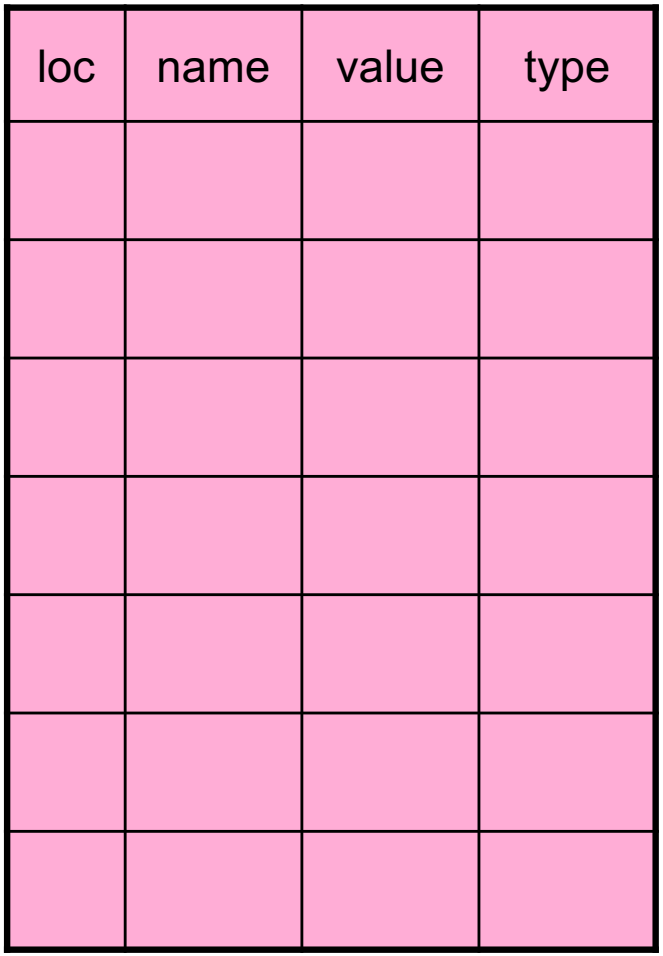
Your iMac uses one of synchronous dynamic without wasting clock

- 4GB 1066MHz DDR
- 8GB 1066MHz DDR
- 8GB 1066MHz DDR
- 16GB 1066MHz DD



Variables and memory in C++

Stack memory



Pointers - Intro

```
int x;
int * p;
```

How do we assign to p?

```
p =
```

```
p =
```

_____ operator: &

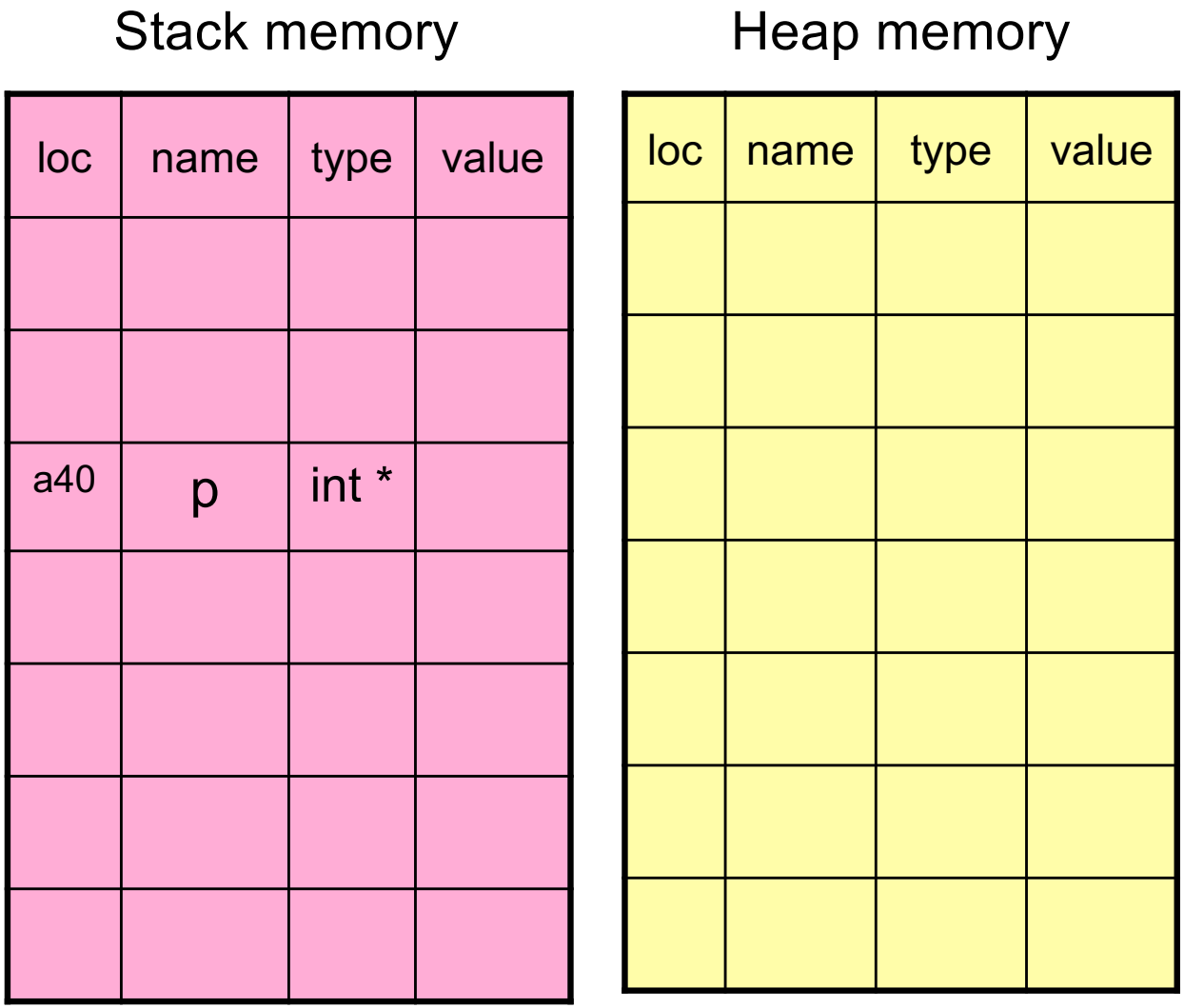
_____ operator: *

Stack memory

loc	name	value	type
a20	x	5	int
a40	p		int *

Pointer variables and dynamic memory allocation:

```
int * p;
```



Youtube: pointer binky c++

Fun and games with pointers: (warm-up)

```
int * p, q;
```

What type is q?_____

```
int *p;
int x;
p = &x;
*p = 6;
cout << x;
cout << p;
```

What is output?_____

What is output?_____

Write a statement whose output is the value of `x`, using variable `p`: _____

```
int *p, *q;
p = new int;
q = p;
*q = 8;
cout << *p;
q = new int;
*q = 9;
p = NULL;
delete q;
q = NULL;
```

What is output?_____

Do you like this?_____

Do you like this?_____

Memory leak:

Deleting a null pointer:

Dereferencing a null pointer:

Fun and games with pointers:

```
int * p, * q;
p = new int;
q = p;
delete p;
… // some random stuff
cout << *q;
```

Do you like this?_____

Stack vs. Heap memory:

```
void fun() {
  string s = "hello!";
  cout << s << endl;
}
int main() {
  fun();
  return 0;
}
```

System allocates space for s and takes care of freeing it when s goes out of scope.

Data can be accessed directly, rather than via a pointer.

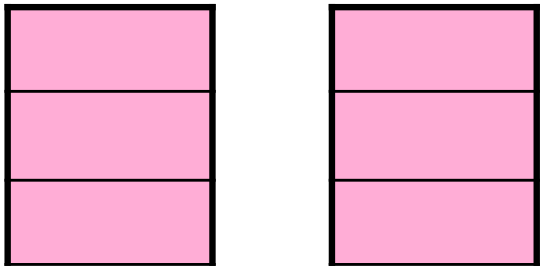
```
void fun() {
  string * s = new string;
  *s = "hello?";
  cout << *s << endl;
  delete s;
}
int main() {
  fun();
  return 0;
}
```

Allocated memory must be deleted programmatically.

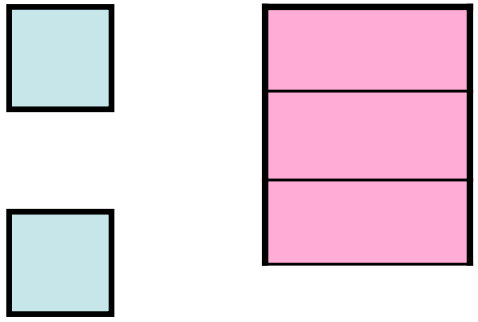
Data must be accessed by a pointer.

Pointers and objects:

```
face a, b;
… // init b
a = b;
a.setName(“ann”);
b.getName();
```



```
face * c, * d;
… // init *d
c = d;
c->setName(“carlos”);
(*d).getName();
```



```
class face {
public:
     void setName(string n);
     string getName();
     …
private:
     string name;
     PNG pic;
     boolean done;
};
```

Practice--

```
int * p; int x;

p = x;
```

Do you like this?_____

What kind of error? Compiler Runtime

```
int * p, * q;

p = new int;

q = p;

delete p;

… // some random stuff

cout << *q;
```

Do you like this?_____

```
int * p;

*p = 37;

p = NULL;

*p = 73;
```

Do you like this?_____

What kind of error? Compiler Runtime

```
int * p;  int x;
```

Variable `p` can be given a target (pointee) in two ways. Write an example of each.

Use the letters S and H in a meaningful way to tell where the pointee exists in memory.

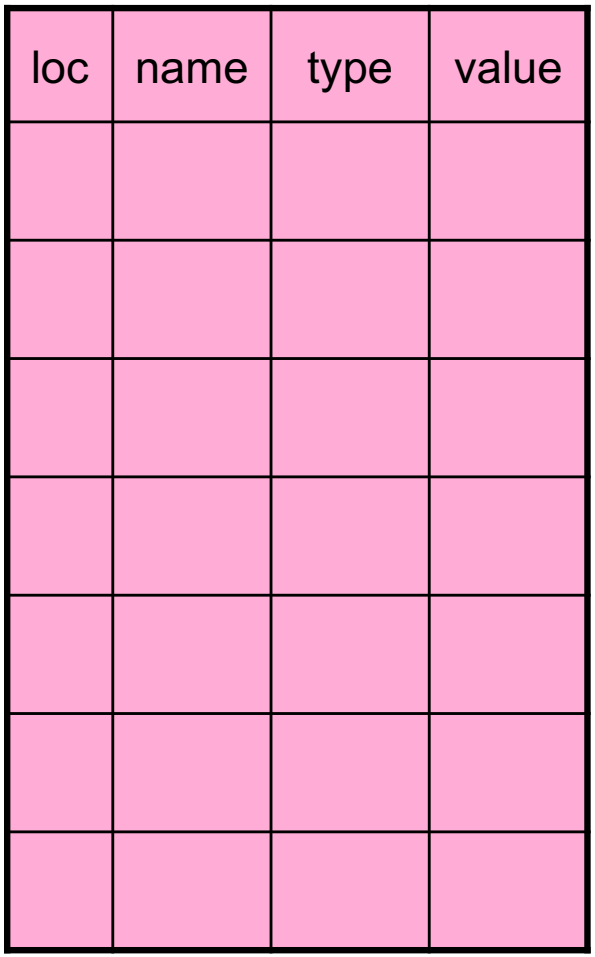
C++ Short Course Part 1

- Classes
- Pointers
- Arrays
- Parameter passing
- Return values

Arrays: static (stackic)

```
int x[5];
```

Stack memory



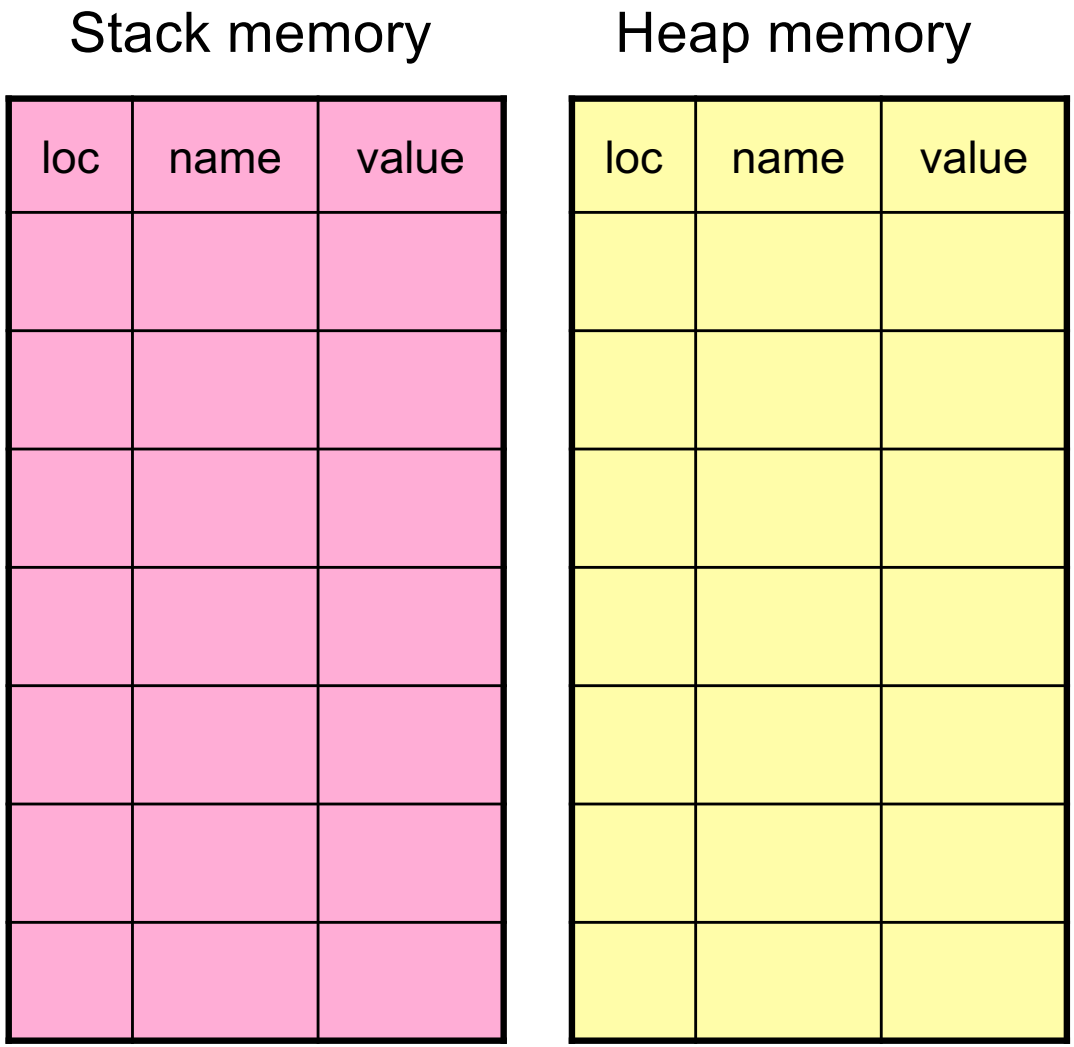
Arrays: dynamic (heap)

```
int * x;


int size = 3;
x = new int[size];


for(int i=0, i<size, i++)
    x[i] = i + 3;


delete [] x;
```



A point to ponder: How is my `garden` implemented?

```
class garden{
public:
…
// all the public members
…
private:
   flower ** plot;
   // other stuff
};
```

Option 1:

Option 2:

Option 3:

Option 4:

C++ Short Course Part 1

- Classes
- Pointers
- Arrays
- Parameter passing
- Return values

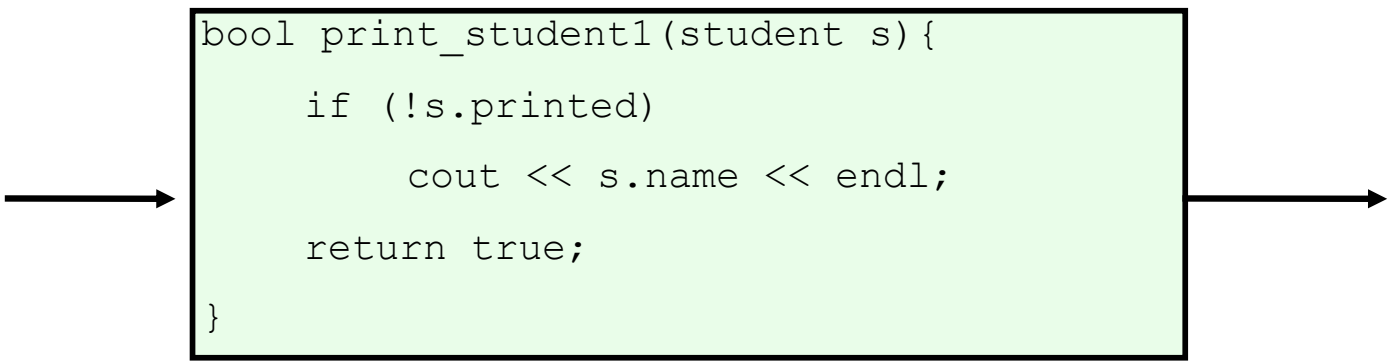
Parameter passing:

```
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);
}
```

?



Parameter passing:

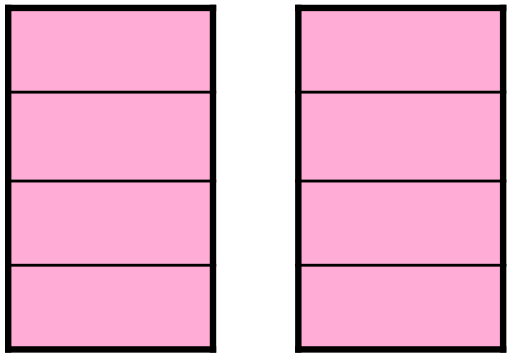
```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

Function defn

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

Example of use

```
student a;
… // initialize a
a.printed = print_student1(a);
cout << a.printed << endl;
```



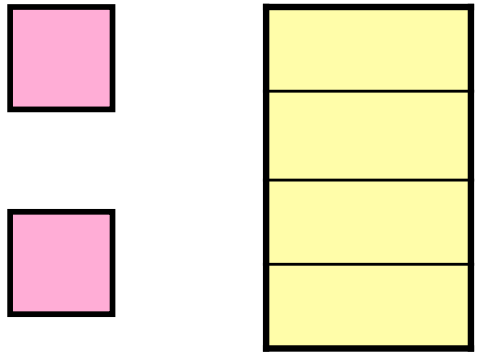
Parameter passing:

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

Function defn

```
void print_student2(student    s){
    if (!  s.printed)
        cout <<   s.name << endl;

}
```



Example of use

```
student * b;
… // initialize b
print_student2(b);
cout << b.printed << endl;
```

Parameter passing:

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

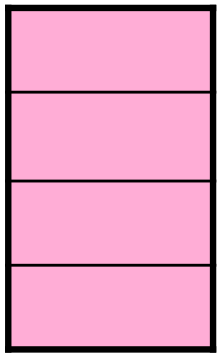
Function defn

```
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;
```



C++ Short Course Part 1

- Classes
- Pointers
- Arrays
- Parameter passing
- Return values

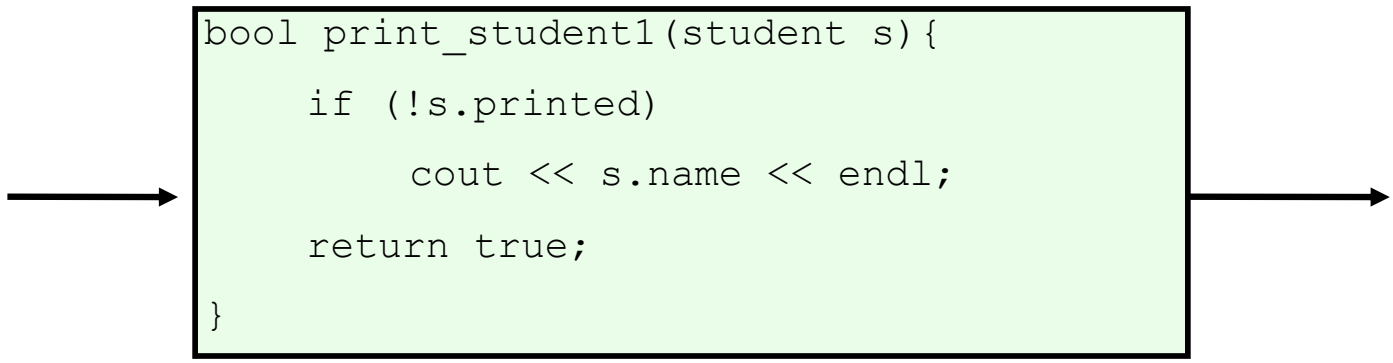
```
struct student {
    string name;
    PNG mug;
    bool printed; // print flag
};
```

Return values:

What happens when we run code like this:

```
int main() {
    student a;
    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.