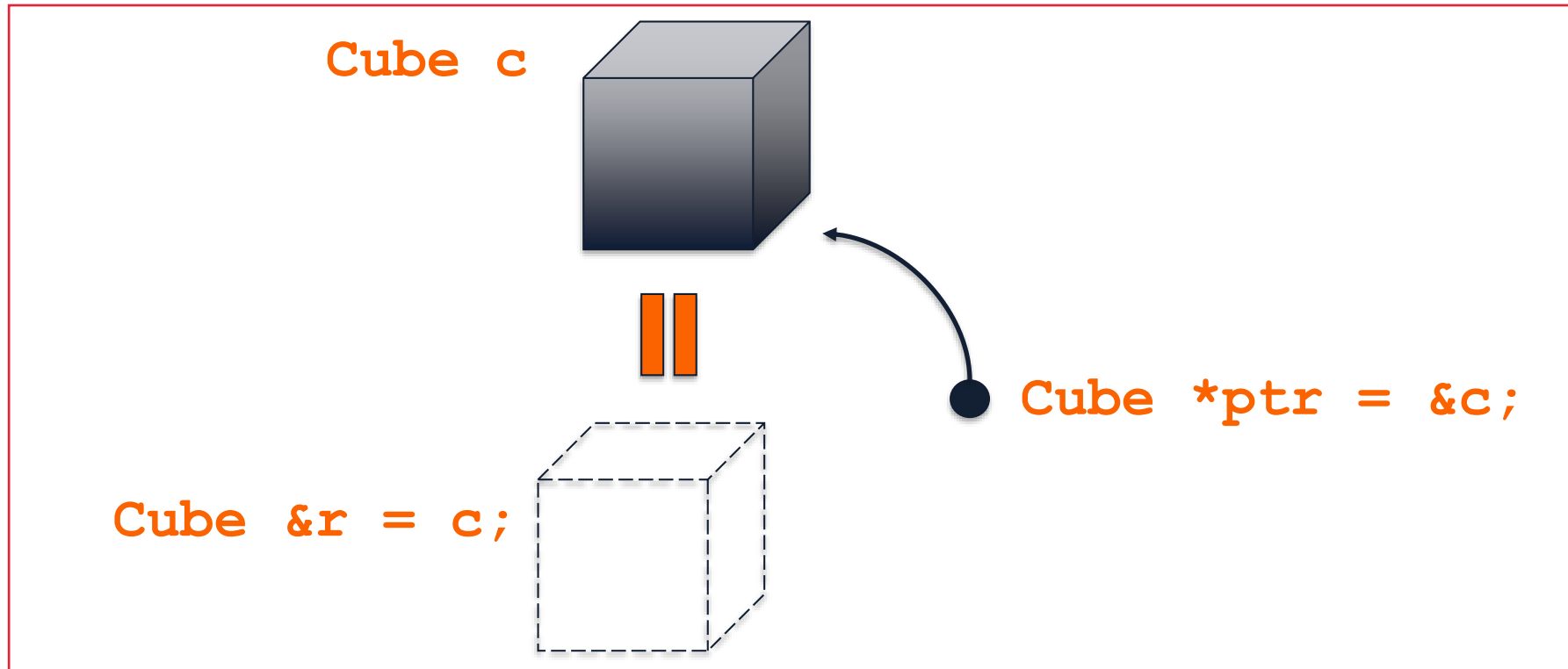


Variable Storage

Prof. Wade Fagen-Ulmschneider

I ILLINOIS

In C++, an instance of a variable can be stored directly in memory, accessed by pointer, or accessed by reference.



Direct Storage

By default, variables are stored directly in memory.

- The **type** of a variable has no modifiers.
- The object takes up exactly its size in memory.

```
Cube c;           // Stores a Cube in memory  
int i;           // Stores an integer in memory  
uiuc::HSLAPixel p; // Stores a pixel in memory
```

Storage by Pointer

- The **type** of a variable is modified with an asterisk (*).
- A pointer takes a “memory address width” of memory (ex: 64 bits on a 64-bit system).
- The pointer “points” to the allocated space of the object.

```
Cube *c;           // Pointer to a Cube in memory  
int *i;            // Pointer to an integer in memory  
uiuc::HSLAPixel *p; // Pointer to a pixel in memory
```

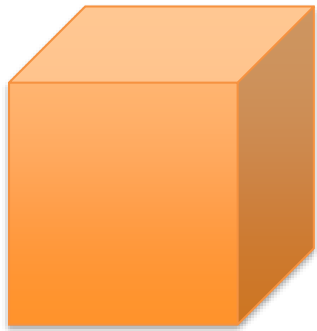
Storage by Reference

- A reference is an **alias** to existing memory and is denoted in the type with an ampersand (&).
- A reference does not store memory itself, it is only an alias to another variable.
- The alias must be assigned when the variable is initialized.

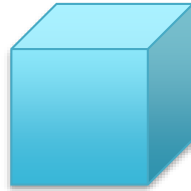
```
Cube &c = cube;           // Alias to the variable `cube`  
int &i = count;           // Alias to the variable `i`  
uiuc::HSLAPixel &p;      // Illegal! Must alias something  
                           when variable is initialized.
```

Example: Cube Currency

Suppose our cubes have a value to them, based on their volume:



10^3 volume
💰 1,000



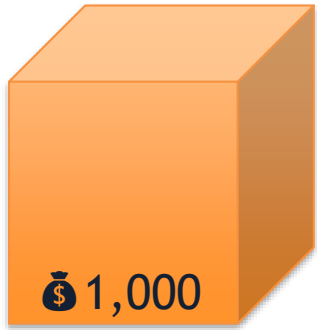
5^3 volume
💰 125



1^3 volume
💰 1

Example: Cube Currency

When we receive money, we want the cube itself - not a copy of the cube.



cpp-memory2/Cube.cpp

```
12 Cube::Cube(double length) {
13     length_ = length;
14     std::cout << "Created $" << getVolume() << std::endl;
15 }
16
17 Cube::Cube(const Cube & obj) {
18     length_ = obj.length_;
19     std::cout << "Created $" << getVolume() << " via copy" << std::endl;
20 }
21
22 Cube & Cube::operator=(const Cube & obj) {
23     std::cout << "Transformed $" << getVolume() << "-> $" <<
24                                     obj.getVolume() << std::endl;
25     length_ = obj.length_;
26     return *this;
27 }
```


cpp-memory2/ex1/byValue.cpp

```
11 int main() {  
12     // Create a 1,000-valued cube  
13     Cube c(10);  
14  
15     // Transfer the cube  
16     Cube myCube = c;  
17  
18     return 0;  
19 }
```

← copy constructor creates
another obj

cpp-memory2/ex1/byRef.cpp

```
11 int main() {  
12     // Create a 1,000-valued cube  
13     Cube c(10);  
14  
15     // Transfer the cube  
16     Cube & myCube = c;  
17  
18     return 0;  
19 }
```

transfer by reference, NOT create
another obj so nothing happend

cpp-memory2/ex1/byPointer.cpp

```
11 int main() {  
12     // Create a 1,000-valued cube  
13     Cube c(10);  
14  
15     // Transfer the cube  
16     Cube * myCube = &c;  
17  
18     return 0;  
19 }
```

transfer by pointer, a new stack variable.
There are two variables point to the same
obj, a variable and a pointer

Pass by _____

Identical to storage, arguments can be passed to functions in three different ways:

- Pass by **value** (default)
- Pass by **pointer** (modified with *****)
- Pass by **reference** (modified with **&**, acts as an alias)

cpp-memory2/ex2/byValue.cpp


```
11 bool sendCube(Cube c) {
12     // ... logic to send a Cube somewhere ...
13     return true;
14 }
15
16 int main() {
17     // Create a 1,000-valued cube
18     Cube c(10);
19
20     // Send the cube to someone
21     sendCube(c);
22
23     return 0;
24 }
```

copy constructor creates
another obj



cpp-memory2/ex2/byRef.cpp


```
11 bool sendCube(Cube & c) {  
12     // ... logic to send a Cube somewhere ...  
13     return true;  
14 }  
15  
16 int main() {  
17     // Create a 1,000-valued cube  
18     Cube c(10);  
19  
20     // Send the cube to someone  
21     sendCube(c);  
22  
23     return 0;  
24 }
```



send by reference, NOT create another obj

cpp-memory2/ex2/byPointer.cpp

```
11 bool sendCube(Cube * c) {
12     // ... logic to send a Cube somewhere ...
13     return true;
14 }
15
16 int main() {
17     // Create a 1,000-valued cube
18     Cube c(10);
19
20     // Send the cube to someone
21     sendCube(&c);
22
23     return 0;
24 }
```



send by pointer, NOT create another obj

Return by _____

Similarly, values can be returned all three ways as well:

- Return by **value** (default)
- Return by **pointer** (modified with *****)
- Return by **reference** (modified with **&**, acts as an alias)
 - *Never return a reference to a stack variable created on the stack of your current function!*