References in C++

- Definition *Reference:– An Alternative Name for an Object*

BIG difference from Java

- *References* are only created in declarations and parameters
- A reference can only appear where the object itself could have appeared

Simple References

```
void f() {
  int j = 1;
  int &r = j;   //r and j refer to same int
  int x = r;    // x now is 1

  r = 2;        // j now is 2
} //f
```

Sometimes, reference declarations are written as

`int& r1 = k`

```
int k;
int &r1 = k;    // okay: r1 is initialized
int &r2;        // error; initializer missing
extern int &r3;      //okay; r3 defined
  elsewhere
```

Simple References (continued)

```
void g() {
  int ii = 0;
  int &rr = ii;
  rr++;          // ii now is 1

  int *pp = &rr;     // pp now points to ii
} //g
```

These `'*'` and `'&'` are *not* operators but rather declaration qualifiers.

This `'&'` is the unary address operator brought over from *C*.

Note: This declares a pointer exactly as in *C*, and initializes it with the *address* of **`rr`** (which is another name for **`ii`**)

Reference Parameters

- An *alias* for its corresponding argument in a function call.
  - & placed after the parameter type in the function prototype and function header
- Example
  - int &count in a function header
    - Pronounced as "count is a reference to an int"
- Parameter name in the called function body actually refers to the original variable in the calling function.

Reference Parameter Example

- *C* version

```
void swap (int *a, int *b) {
  int temp = *a;
  *a = *b;
  *b = temp;
} //     void swap(…)
```

Hazard: a **`NULL`** pointer

- *C++* version

```
void swap (int &a, int &b) {
  int temp = a;
  a = b;
  b = temp;
} //     void swap(…)
```

Non-hazard: no pointer here

Notes on References and Pointers

- Pointers in *C* do multiple duty
  - *Links*, as in linked lists and trees
  - *Parameters*, where the function needs to return a value to an argument provided by the caller
  - *Short-hand*, a short way of referring to an object that otherwise would need a complex expression
  - …

Java *vs.* *C*++ References

- In *Java*, a reference is a data type.
    - It can be assigned to, compared, copied, stored, etc.
    - Same reference can refer to different objects at different times during execution
- In *C*++, a reference is an *alias* for an object
    - It cannot be assigned to; assignment is *through* the reference to the underlying object
        - Similar to dereferencing a pointer in *C*
    - A reference *always* refers to the same object for the duration of its scope

Repeat Three Times

A reference is not a *pointer,* ...

A reference is *not* a pointer, …

A reference *is* not a pointer, …

And neither of them resembles a Java reference

Questions?