

# CSE 333 – SECTION 4

---

References, const and classes

# HW2

- Online now. Due on Tuesday, May 6<sup>th</sup> by 11pm.
- Take a look at it soon. Start early.
- File crawler and Indexer.

# This or that?

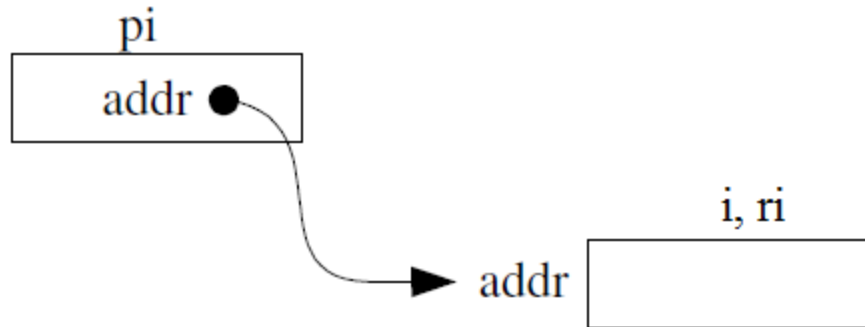
- Consider the following code:

## Pointers:

```
int i;  
int *pi = &i;
```

## References:

```
int i;  
int &ri = i;
```



They are used differently in expressions:

```
*pi = 4;
```

```
ri = 4;
```

# Pointers and References

- Once a reference is created, it cannot be later made to reference another object.
  - Compare to pointers, which are often reassigned.
- References cannot be *null*, whereas pointers can.
- References can never be uninitialized. It is also impossible to reinitialize a reference.

# C++ const declaration

- As a declaration specifier, const changes the type to make the data bits unmodifiable.

```
const int m = 255;
```

- Pointer/Reference to constant integer:

```
int n = 100;
```

```
const int *pi = &n; // *pi is read only
```

```
const int &ri = n; // ri is read only
```

# When to use?

- Function parameter types and return types and functions that declare overloaded operators.
- **Pointers:** may point to many different objects during its lifetime. Pointer arithmetic (++ or --) enables moving from one address to another. (Arrays, for e.g.)
- **References:** can refer to only one object during its lifetime.
- **Style Guide Tip:**
  - use const reference parameters to pass input
  - use pointers to pass output parameters
  - input parameters first, then output parameters last

# C++ Classes

```
/* Note: This code is unfinished! Beware! */  
class Point {  
public:  
    Point(const int x, const int y); // constructor  
    int get_x() { return x_; } // inline member function  
    int get_y() { return y_; } // inline member function  
    double distance(const Point &p); // member function  
    void setLocation(const int x, const int y); //member function  
private:  
    int x_; // data member  
    int y_; // data member  
}; // class Point
```

# Section Exercise – Part I

- Define a class Rectangle whose instance variables are a pair of Point objects (upper left, lower right).
- Include at least one constructor. Make sure you get const right in the right places.
- Methods:
  - **getul(), getlr()** - returns upper and lower points.
  - **cornerPoints()** – to obtain the corner points.
  - **area()** - returns the Rectangle's area.
  - **contains(Point &p)** - returns true or false depending on whether point p is inside the rectangle.