CSE 333 – SECTION 4

References, const and classes

HW2

- Index the contents of files
- Search through documents containing specified words
- Feels good when you complete it

This or that?

- Consider the following code:

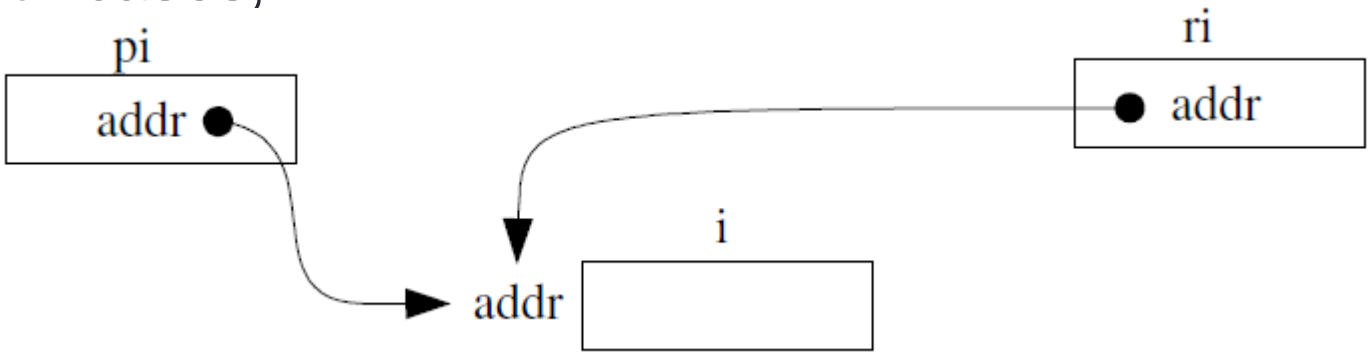
Pointers:

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

References:

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

In both cases,



The difference lies in how they are used 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 is a type specifier that makes objects unmodifiable.

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

- Reference to constant integer:

```
int n = 100;
const int &ri = n; //ri becomes 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
```