

# CSE 333 – SECTION 4

C++ Classes, Const and References;

*Some slides referenced from CSE 333 -Winter 2018 slides*

# Logistics

Friday (tomorrow)

Exercise 9 @ 10 am

Monday

Exercise 10 @ 10 am

Wednesday:

Exercise 11 @ 10 am

Thursday (1 week from now):

HW2 @ 11 pm

# Section Plan

- C++ const/reference refresher
- References Problem #2
- Mult-Choice

# Example

Similar in syntax to the \*  
in pointer declarations

- Consider the following code:

```
int x = 5;
```

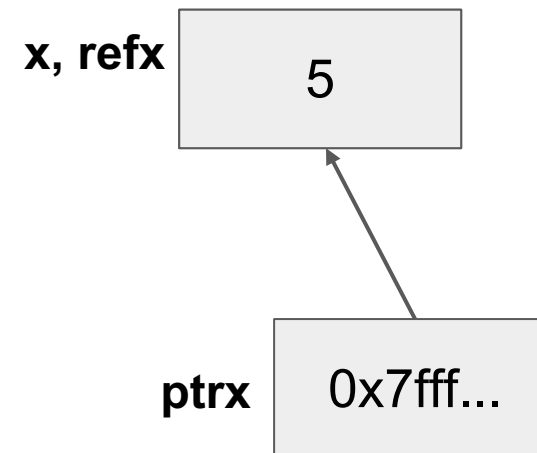
```
int &refx = x;
```

```
int *ptrx = &x;
```

Difference between declaring reference and 'address of' operator.

& in type declaration is for reference declaration.  
Elsewhere it is 'address of'

What are some tradeoffs to using pointers vs references?



## Legend

**Red Thing** = "can't change the box it's next to"

**Black** = "writeable/readable"

# Summary

- Pointers vs. References:

<b>Pointers</b>	<b>References</b>
Can move to different data via reassignment/pointer arithmetic	References the same data for its entire lifetime
Can be initialized to NULL	No sensible "default reference"
Used for output parameters e.g. <code>MyClass*</code> output	Used for input parameters e.g. <code>const MyClass&amp;</code> input

## When would you prefer reference to pointer as function parameters?

- When you don't want to deal with pointer semantics, use references
- When you don't want to copy stuff over (doesn't create a copy, especially for parameters and/or return values), use references
- Style wise, we want to use **references for input parameters** and **pointers for output parameters**, with the output parameters declared last

# Const

- Mark a variable with `const` to make a compile time check that a variable is never reassigned
- Does not change the underlying write-permissions for this variable

```
int x = 42;
```

```
const int* ro_ptr = &x; // Read only
```

```
int* rw_ptr = &x; // x can still be modified with rw_ptr!
```

```
int* const ptr = &x; // Only ever points to x
```

# Example

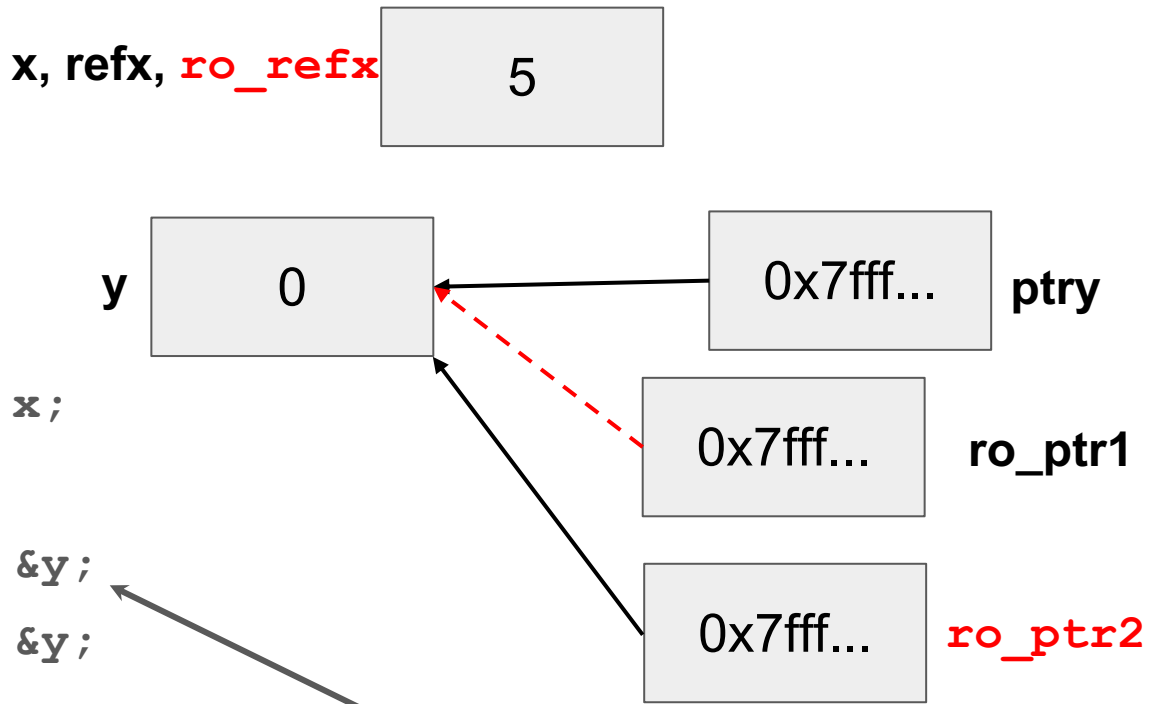
- Consider the following code:

```
int x = 5, y = 0;  
int &refx = x;  
const int &ro_refx = x;  
int *ptry = &y;  
const int *ro_ptr1 = &y;  
int *const ro_ptr2 = &y;
```

“Const pointer to an int”

“Pointer to a const int”

**Tip:** Read the declaration “right-to-left”

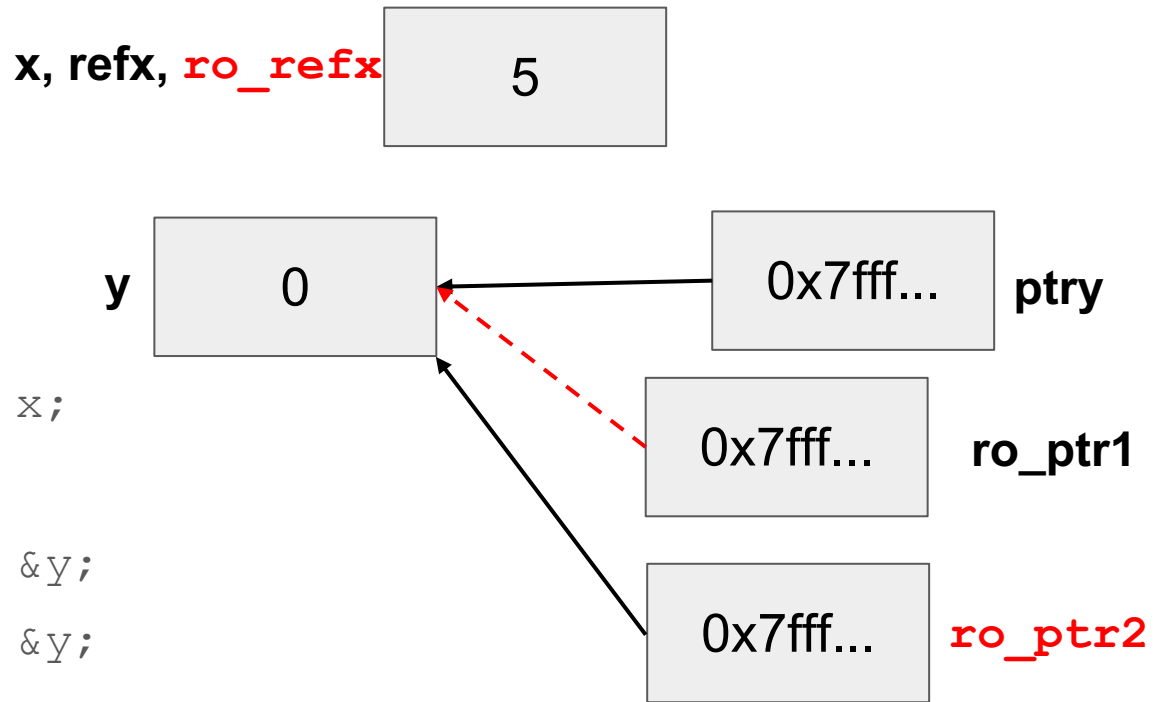


**Legend**  
**Red Thing** = “can’t change the box it’s next to”  
**Black** = “writeable/readable”

# Example

- Consider the following code:

```
int x = 5, y = 0;  
int &refx = x;  
const int &ro_refx = x;  
int *ptry = &y;  
const int *ro_ptr1 = &y;  
int *const ro_ptr2 = &y;
```



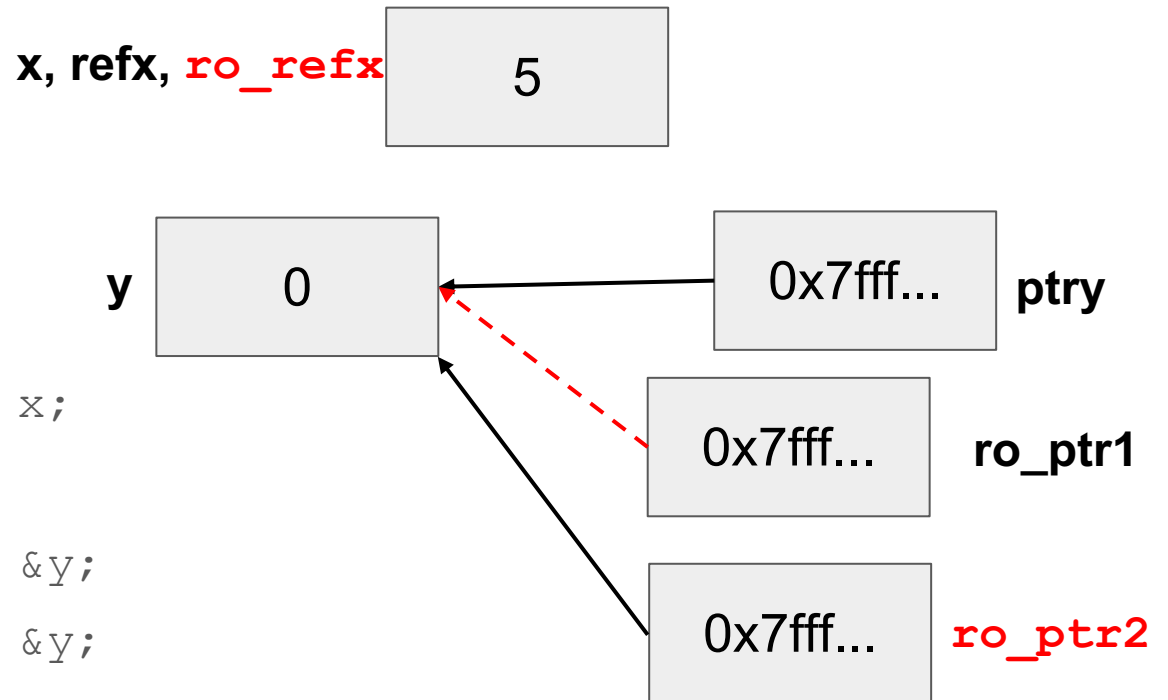
**Tip:** Read the declaration “right-to-left”

**Legend**  
**Red Thing** = “can’t change the box it’s next to”  
**Black** = “writeable/readable”

# Example

- Consider the following code:

```
int x = 5, y = 0;  
int &refx = x;  
const int &ro_refx = x;  
int *ptry = &y;  
const int *ro_ptr1 = &y;  
int *const ro_ptr2 = &y;
```



**Which results in a compiler error?**

```
ok   bar(refx);  
Error bar(ro_refx);  
ok   foo(refx);
```

**Legend**

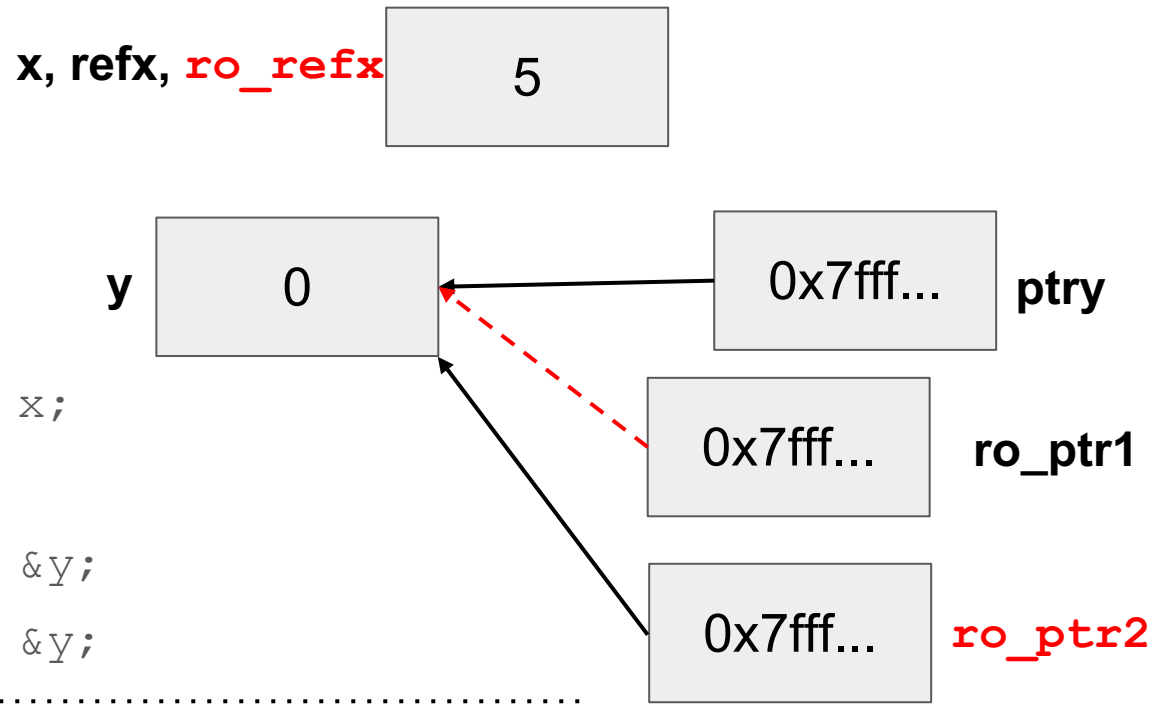
**Red Thing** = "can't change the box it's next to"

**Black** = "writeable/readable"

# Example

- Consider the following code:

```
int x = 5, y = 0;
int &refx = x;
const int &ro_refx = x;
int *ptry = &y;
const int *ro_ptr1 = &y;
int *const ro_ptr2 = &y;
```



**Which of these results in a compiler error?**

- ok** `ro_ptr1 = (int*)0xDEADBEEF;`
- Error** `ro_ptr2 = ro_ptr2 + 2;`
- Error** `*ro_ptr1 = *ro_ptr1 + 1;`

**Legend**

- Red Thing** = "can't change the box it's next to"
- Black** = "writeable/readable"

# C++ Classes

# Class Organization

Point.h

```
class Point {  
public:  
    Point(int x, int y);  
    int get_x() { return x_; }  
    int get_y() { return y_; }  
    double Distance(Point & p);  
    void SetLocation(int x, int y);  
private:  
    int x_;  
    int y_;  
};
```

Class declaration goes in Point.h,  
implementation goes in Point.cc.

Point.cc

```
Point::Point(int x, int y){  
    x_ = x;  
    this->y_ = y;  
}  
  
double Point::Distance(Point &p){  
    double xdiff = pow(x_ - p.x_, 2);  
    double ydiff = pow(y_ - p.y_, 2);  
    return sqrt(xdiff + ydiff);  
}  
  
void Point::SetLocation(int x, int y){  
    x_ = x;  
    this->y_ = y;  
}
```

# Class .h files

Point.h

```
class Point {  
    public:  
        Point(int x, int y);  
        int get_x() { return x_; }  
        int get_y() { return y_; }  
        double Distance(Point & p);  
        void SetLocation(int x, int y);  
    private:  
        int x_;  
        int y_;  
};
```

- Includes the class declaration.
- Can specify member functions and variables and whether they are public/private
- Can have implementation of functions, usually only done with simple functions (e.g. getters)

# Class .cc files

Contains member function definitions.  
These are indicated by:  
Class\_Name::Func\_name(){

If not specified as part of the class, it cannot access private class members, and probably won't compile.

Point.cc

```
Point::Point(int x, int y){  
    x_ = x;  
    this->y_ = y;  
}  
  
double Point::Distance(Point &p){  
    double xdiff = pow(x_ - p.x_, 2);  
    double ydiff = pow(y_ - p.y_, 2);  
    return sqrt(xdiff + ydiff);  
}  
  
void Point::SetLocation(int x, int y){  
    x_ = x;  
    this->y_ = y;  
}
```

What about “const” object methods?

```
#ifndef POINT_H_
#define POINT_H_

class Point {
public:
    Point(const int x, const int y);
    int get_x() const { return x_; }
    int get_y() const { return y_; }
    double Distance(const Point& p) const;
    void SetLocation(const int x, const int y);

private:
    int x_;
    int y_;
}; // class Point

#endif
```

Cannot mutate the object it's called on.  
**Trying to change x\_ or y\_ inside will cause a compiler error!**

## Exercise 3

```
class MultChoice {
public:
    MultChoice(int q, char resp) : q_(q), resp_(resp) { } // 2-arg ctor
    int get_q() const { return q_; }
    char get_resp() { return resp_; }
    bool Compare(MultChoice &mc) const; // do these MultChoice's match?

private:
    int q_; // question number
    char resp_; // response: 'A','B','C','D', or 'E'
}; // class MultChoice
```

Code	Error?	Code	Error?
<pre>int z = 5; const int *x = &amp;z; int *y = &amp;z; x = y; *x = *y;</pre>		<pre>int z = 5; int *const w = &amp;z; const int *const v = &amp;z; *v = *w; *w = *v;</pre>	

# Exercise 3

```
class MultChoice {
public:
    MultChoice(int q, char resp) : q_(q), resp_(resp) { } // 2-arg ctor
    int get_q() const { return q_; }
    char get_resp() { return resp_; }
    bool Compare(MultChoice &mc) const; // do these MultChoice's match?

private:
    int q_; // question number
    char resp_; // response: 'A','B','C','D', or 'E'
}; // class MultChoice
```

Code	Error?	Code	Error?
<pre>int z = 5; const int *x = &amp;z; int *y = &amp;z; x = y; *x = *y;</pre>	<p><b>N</b> <b>N</b> <b>N</b> <b>N</b> <b>Y</b></p>	<pre>int z = 5; int *const w = &amp;z; const int *const v = &amp;z; *v = *w; *w = *v;</pre>	<p><b>N</b> <b>N</b> <b>N</b> <b>Y</b> <b>N</b></p>

## Exercise 3

```
class MultChoice {
public:
    MultChoice(int q, char resp) : q_(q), resp_(resp) { } // 2-arg ctor
    int get_q() const { return q_; }
    char get_resp() { return resp_; }
    bool Compare(MultChoice &mc) const; // do these MultChoice's match?

private:
    int q_; // question number
    char resp_; // response: 'A','B','C','D', or 'E'
}; // class MultChoice
```

Code	Error?	Code	Error?
<pre>const MultChoice m1(1, 'A'); MultChoice m2(2, 'B'); cout &lt;&lt; m1.<b>get_resp</b>(); cout &lt;&lt; m2.<b>get_q</b>();</pre>		<pre>const MultChoice m1(1, 'A'); MultChoice m2(2, 'B'); m1.<b>Compare</b>(m2); m2.<b>Compare</b>(m1);</pre>	

## Exercise 3

```
class MultChoice {
public:
    MultChoice(int q, char resp) : q_(q), resp_(resp) { } // 2-arg ctor
    int get_q() const { return q_; }
    char get_resp() { return resp_; }
    bool Compare(MultChoice &mc) const; // do these MultChoice's match?

private:
    int q_; // question number
    char resp_; // response: 'A','B','C','D', or 'E'
}; // class MultChoice
```

Code	Error?	Code	Error?
const MultChoice m1(1, 'A'); MultChoice m2(2, 'B'); cout << m1. <b>get_resp</b> (); cout << m2. <b>get_q</b> ();	<b>N</b> <b>N</b> <b>Y</b> <b>N</b>	const MultChoice m1(1, 'A'); MultChoice m2(2, 'B'); m1. <b>Compare</b> (m2); m2. <b>Compare</b> (m1);	<b>N</b> <b>N</b> <b>N</b> <b>Y</b>

What would you change about the class declaration to make it better?

```
class MultChoice {
public:
    MultChoice(int q, char resp) : q_(q), resp_(resp) { } // 2-arg ctor
    int get_q() const { return q_; }
    char get_resp() { return resp_; }
    bool Compare(MultChoice &mc) const; // do these MultChoice's match?

private:
    int q_; // question number
    char resp_; // response: 'A','B','C','D', or 'E'
}; // class MultChoice
```