this Pointer and Dynamic Memory Management

Cascading Member Function Calls

- Let's examine fig. 7.14 - 7.16

Dynamic Memory Management

- In C++, programmers can control the allocation and deallocation of memory for any user-defined or built-in data type
- The operators used are:
  - new
  - delete
- Must also have
  - #include <new>

Example

```
Time *pTime1;
pTime1 = new Time;
pTime1->print();

Time *pTime2;
pTime2 = new Time( 15, 2, 59 );
pTime2->print();

Time t( 2, 23, 90 );
Time *pTime3 = &t;
pTime3->print();
```


Example

- What would happen if we output the values of each of the pointers on the previous slides
  - `cout << pTime1 << endl;`
- How would we free the space pointed to by the pointers

Problem

- Create a class called **`IntegerSet`** where a set is represented as an array of ones and zeros. Array element **`a[i]`** is **`1`** if integer **`i`** is in the set and array element **`a[j]`** is **`0`** if integer **`j`** is not in the set
- An object of type **`IntegerSet`** is instantiated by passing to the constructor an integer representing the range of the set.
  - `IntegerSet mySet( 100 );`

integerset.h

```
class IntegerSet {
public:
   IntegerSet( int );
   IntegerSet( const IntegerSet& );
   IntegerSet unionOfIntegerSets( const IntegerSet& );
   IntegerSet intersectionOfIntegerSets( const IntegerSet& );
   void emptySet();
   void inputSet();
   void insertElement( int );
   void deleteElement( int );
   void setPrint() const;
   bool isEqualTo( const IntegerSet& ) const;
private:
   int *set;  // dynamically allocated set
   int size;
   // function validEntry definition
   bool validEntry( int x ) const
   {
      return x >= 0 && x < size;
   }
};
```


Summary

- Today we covered
  - this Pointer
  - Dynamic memory management
- Completed pages 489 - 497