Lecture 10

Const qualified pointer

- A const-qualified pointer guarantees that the program has read-only access to the data referenced by the pointer

```
void MySub( const int * A )
{
  *A = 50;    // error
  A++;        // ok
}
```

The pointer itself can be modified

Constant pointer

Declaring a constant pointer guarantees only that the pointer itself cannot be modified.

```
void MySub( int * const A )
{
  *A = 50;    // ok
  A++;        // error
}
```

The data referenced by the pointer can still be modified.

Memory allocation of members

- While objects conceptually contain data members and functions, C++ objects typically contain only data.

- The compiler creates only one copy of the class's member functions and shares that copy amongst all the members.

Assignment

- How do member functions know which object's data members to manipulate?

this pointer

- Every object has access to its own address through a pointer called *this* (C++ keyword)

- An object's this pointer is not part of object itself, rather it is passed as an implicit argument to each of the object's member functions

Example

```
class test
{   private: int x;
    public: test(int k=0) : x(k) { }
    void print()
    { cout<<x; //implicit call to function
    cout<<this->x ;}
}
```

```
void main()
{
   test t1;
   t1.print();
}
```

Using this pointer to enable cascaded function calls

```
class test
{   private: int x; int y;
    public: test(int k=0) :
       x(k) { }
    test & setx()
    { x=10;
    return *this; }
    Test & sety()
    { y=20; return *this;}
}
```

```
void main()
{
   test t1;
   t1.setx().sety();
}
```

static class members

- If a data member is static, only one copy of that variable is shared by all objects of a class
- static data members are defined and initialized at file scope

Example

```
class test
{   private:
      static int x;
      int y;
    public:
    void display()
    { cout<<x; x++; }
}
int test::x=0;
```

```
void main()
{
   test t1;
   t1.display();
   test t2;
   t2.display();
}
```

Assignment

- When does a this pointer get created?
- Does the expression *delete p* delete the pointer or the object being pointed to by p?
- The object created using new does not get destroyed when the control returns from the function in which it was created (True/False)