CS 1575 - Data Structures

Notes on Pointers and Classes

Pointers to Classes

Pointers can be used with classes like any other datatype.

Example:

```
class MyClass
{
  int x;
}
…
MyClass *p;           // a pointer to class MyClass
p = new MyClass;      // dynamically allocated class
```

De-referencing a pointer to a class

`p -> x`
Accesses the `x` member of the class type pointed by `p`.

The `*` operator can also be used, but due to its low preference usually requires parenthesis to use correctly.

Example:

```
MyClass *p;
p = new MyClass;
p -> x = 5;           // assigns 5 to member x
(*p).x = 6;           // assigns 6 to member x
*p.x = 7;             // ERROR: p.x is not a pointer
```

The ‘this’ pointer

Inside every non-static member functions, the variable:
`T* const this;`

holds the address of the class object from which the member function was invoked.

Example:

```
class MyClass
{
  int x;
  void foo () {
    this -> x = 42  // using ‘this’ explicitly to assign a value to x
  }
}
```

Classes with Pointer members

C++ automatically generates 3 member functions methods for every class. These are the *destructor,* the *copy constructor*, and the *operator*=.

The Destructor
Called automatically by C++ when a class goes out of scope or is deallocated with `delete`.

The Copy Constructor
Called when:

1. declaration with initialization.
   `MyClass B = A;`
   `MyClass B ( A );`

2. when an object is passed by value (instead of using `&` or `const &`).
   (which you should **not** do)

3. when an object is returned by value (instead of using `&` or `const &`).
   (which you should **not** do either)

The Operator=
Used when assignment between objects is used
`A = B;`

For a class `MyClass`, the default member functions are:

```
/* DESTRUCTOR */
MyClass::~MyClass()
{
}

/* COPY CONSTRUCTOR */
MyClass::MyClass(const MyClass &rhs) :  x ( rhs.x )
{
}

/* OPERATOR= */
const MyClass & MyClass::operator= ( const MyClass & rhs )
{
  if ( this != &rhs )            // Standard alias test
    x = rhs.s;
  return *this;
}
```

The defaults can cause problems when a class has a pointer member and this member is used for dynamically allocating memory.

Example:

```
class IntBox
{
private:
  int *m_int;
public:
  IntBox ( int initValue = 0 ) {      // constructor
    m_int = new int(initValue);
  }
  int read () const {                      // accessor
    return *myInt;
  }
  void write ( int x ) {                   // mutator
```

```
    *m_int = x;
  }
}

int foo(){
  IntBox a(7)
  IntBox b = a;
  IntBox c;

  c = b;
  a.write( 4 );
  cout << a.read() << " " << b.read() << " " << c.read() << endl;
                                        // all values are '4' !!!

  return 0
}
```

The default *copy constructor* and *operator=* copy **only the pointers, not the memory pointed to**. This is called a "shallow copy", and may not be the functionality you intended.

NOTE! : Whenever you create a class which has pointer members, consider overwriting the default member functions.

END.