ECE2036: Week 3_A – Separating Header and Implementation Files

Reading: Continue to read in D&D Chapter 3 and Begin Chapter 4-5

For today we want to show a more realistic implementation of how your class libraries might be integrated with your client's code. The idea is that we will have a header file that has the "class interface," and we will have a source file that shows the implementation of the class's member functions. This is illustrated in the programs below for our complex number example.

If you put these three files in your unix directory, you can compile AND link them with the following command.

```
g++ ComplexNumber.cc main.cc –o testfile
```

You can also compile them separately THEN link the object files together.

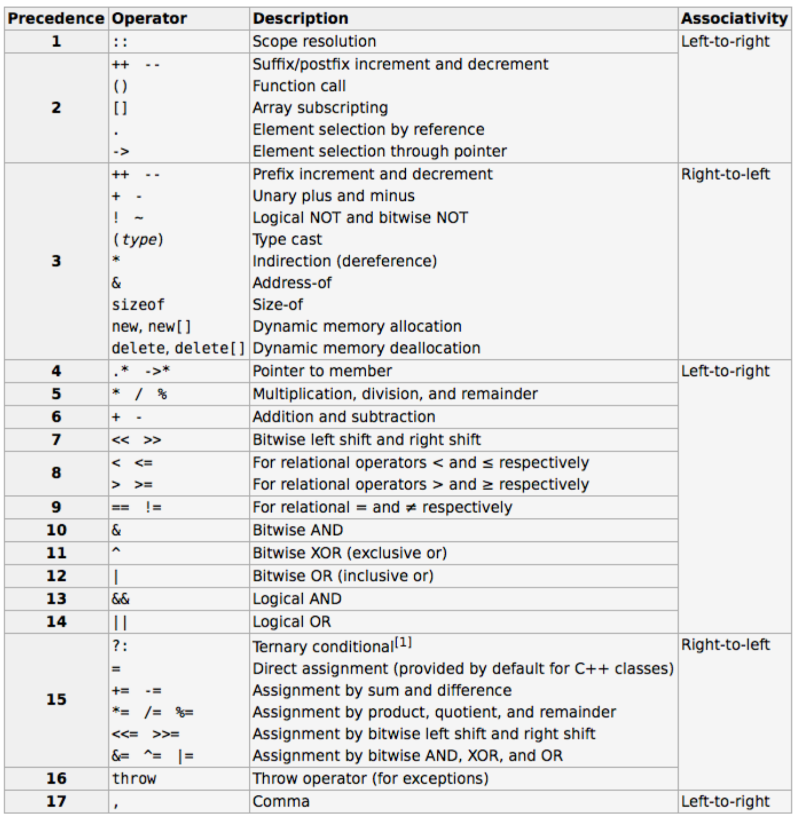
```
g++  ComplexNumber.cc –c
g++ main.cc –c
g++ ComplexNumber.o main.o –o testfile
```

We will also discuss basic ideas in makefiles and sftp to help you with your labs.

```
//This file is called ComplexNumber.h

#include <iostream>
using namespace std;

//THis is an example of a CLASS INTERFACE
class ComplexNum
{

public:  // access specifier

  ComplexNum(float , float ); // specify FUNCTION PROTOTYPES
  ComplexNum(); //notice the semicolon at end
  void setXreal (float);
  void setYimag (float);
  void setComplexNum(float, float);
  float getXreal();
  float getYimag();

private:  //access specifier
  float xreal;
  float yimag;

}; //don't forget the semicolon

//------------------------------------------------------------------------------------------

//This file is called the ComplexNumber.cc
#include <iostream>
#include "ComplexNumber.h"
using namespace std;

ComplexNum::ComplexNum(float xr, float yi)
{
  setComplexNum(xr,yi);
   // Next lines are okay BUT not good programming practice
   //xreal = xr;
   //yimag = yi;
}

ComplexNum::ComplexNum()
{ setComplexNum(0,0);}


void ComplexNum::setXreal (float xr)
```

```
{    xreal = xr; }

void ComplexNum::setYimag (float yi)
{    yimag = yi; }

void ComplexNum::setComplexNum(float xr, float yi)
{
    setXreal(xr);
    setYimag(yi);
}

float ComplexNum::getXreal()
{   return (xreal); }

float ComplexNum::getYimag()
{   return (yimag); }

//------------------------------------------------------------------------------------------

//This file is called main.cc and is representative of the client code
#include <iostream>
#include "ComplexNumber.h"
using namespace std;

///final main function -- like the client code
int main()
{

ComplexNum num1(3,4);
ComplexNum num2;

cout << num1.getXreal() << " +j" << num1.getYimag() <<endl;
cout << num2.getXreal() << " +j" << num2.getYimag() <<endl;

return 0;


}
```

//------------------------ ORDER OF PRECEDENCE CHART --------------------------------------



Precedence	Operator	Description	Associativity
1	`::`	Scope resolution	Left-to-right
2	`++ --`	Suffix/postfix increment and decrement	
	`()`	Function call	
	`[]`	Array subscripting	
	`.`	Element selection by reference	
	`->`	Element selection through pointer	
3	`++ --`	Prefix increment and decrement	Right-to-left
	`+ -`	Unary plus and minus	
	`! ~`	Logical NOT and bitwise NOT	
	`(`*type*`)`	Type cast	
	`*`	Indirection (dereference)	
	`&`	Address-of	
	`sizeof`	Size-of	
	`new`, `new[]`	Dynamic memory allocation	
	`delete`, `delete[]`	Dynamic memory deallocation	
4	`.* ->*`	Pointer to member	Left-to-right
5	`* / %`	Multiplication, division, and remainder	
6	`+ -`	Addition and subtraction	
7	`<< >>`	Bitwise left shift and right shift	
8	`< <=`	For relational operators < and ≤ respectively	
	`> >=`	For relational operators > and ≥ respectively	
9	`== !=`	For relational = and ≠ respectively	
10	`&`	Bitwise AND	
11	`^`	Bitwise XOR (exclusive or)	
12	`|`	Bitwise OR (inclusive or)	
13	`&&`	Logical AND	
14	`||`	Logical OR	
15	`?:`	Ternary conditional[1]	Right-to-left
	`=`	Direct assignment (provided by default for C++ classes)	
	`+= -=`	Assignment by sum and difference	
	`*= /= %=`	Assignment by product, quotient, and remainder	
	`<<= >>=`	Assignment by bitwise left shift and right shift	
	`&= ^= |=`	Assignment by bitwise AND, XOR, and OR	
16	`throw`	Throw operator (for exceptions)	
17	`,`	Comma	Left-to-right