ECE2036: Week2_B – Constructors and Constructor Overloading

Reading: Continue to read in D&D Chapter 3

The new idea on this handout is related how we can declare an object. The term that we will use is "instantiate" an object. We would like to be able to initialize the object as a part of the declaration, and we will do this with a special function called a "constructor."

Constructors are special member functions in classes. They have to have the same name as the class and they have no return type. In addition, we can "overload" constructors such that we can have multiple implementations of the constructor, which we will illustrate in class. (Note later we will show that we can overload other functions as well!!)

```
#include <iostream>
using namespace std;

class ComplexNumber
{
public
  ComplexNumber(float xr, float yi);
  void setReal(float xr);
  void setImag (float yi);
  void setComplexNumber(float xr, float yi);
  float getReal();
  float getImag();

private:
  float real;
  float imag;
};

//Now show the function definitions
ComplexNumber::ComplexNumber(float xr, float yi)
{ setComplexNumber(xr,yi); }
void ComplexNumber::setReal(float xr)
{ real = xr; }
void ComplexNumber::setImag (float yi)
{ imag = yi; }
void ComplexNumber::setComplexNumber(float xr, float yi)
{ // good programming practice to reuse code
setReal(xr);
setImag(yi);
}
float ComplexNumber::getReal()
{ return(real); }
float ComplexNumber::getImag()
{  return(imag); }

int main()
{ //begin main
ComplexNumber num1(3,4);
cout << num1.getReal()  << " +j" << num1.getImag() << endl;
//example of nesting functions
num1.setReal(num1.getImag()); // overwrites the real data
//print result
cout << "The real part of num1 is " << num1.getReal() << endl;
}//end main
```

TERM REVIEW SO FAR

- Machine Language
- Assembly Language
- High-Level Language
- Source Code
- Object Code
- Namespace and scope resolution operator
- using namespace std
- Fundamental data types: bool, char, int, float, double…
- Declaring and initializing variables
- Preprocessor directives: (example so far: #include)
- I/O using cout, cin
- I/O stream manipulators: setw, setfill, hex, oct, dec, fixed, scientific, setprecision
- Global and local variables
- fstream objects – ifstream and ofstream objects
- sequential text file manipulation
- objects
- classes
- member functions : public
- data members : private (or public)
- public and private access specifiers
- set and get member functions
- constructors
- overloading constructors
- instantiating objects