DEFINING MEMBER FUNCTIONS AND DATA HIDING IN CPP

A member operator function takes this general form:

*ret-type class-name::*operator#*(arg-list)*

{

// operations

}

Often, operator functions return an object of the class they operate on, but *ret-type* can be any valid type. The # is a placeholder. When you create an operator function, substitute the operator for the #. For example, if you are overloading the / operator, use **operator/.** When you are overloading a unary operator, *arg-list* will be empty. When you are overloading binary operators, *arg-list* will contain one parameter. (The reasons for this seemingly unusual situation will be made clear in a moment.) Here is a simple first example of operator overloading. This program creates a class called **loc**, which stores longitude and latitude values. It overloads the + operator relative to this class. Examine this program carefully, paying special attention to the definition of **operator+( )**:

```
#include <iostream>
using namespace std;
class loc {
int longitude, latitude;
public:
loc() {}
loc(int lg, int lt) {
longitude = lg;
latitude = lt;
}
void show() {
cout << longitude << " ";
cout << latitude << "\n";
}
loc operator+(loc op2);
};
```

```
// Overload + for loc.
loc loc::operator+(loc op2)
{
loc temp;
temp.longitude = op2.longitude + longitude;
temp.latitude = op2.latitude + latitude;
return temp;
}
int main()
{
loc ob1(10, 20), ob2( 5, 30);
ob1.show(); // displays 10 20
ob2.show(); // displays 5 30
ob1 = ob1 + ob2;
ob1.show(); // displays 15 50
return 0;
}
```

2.6 Data hiding

Data hiding is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse. In an object-oriented language, code and data may be combined in such a way that a self-contained "black box" is created. When code and data are linked together in this fashion, an *object* is created. In other words, an object is the device that supports encapsulation. Within an object, code, data, or both may be *private* to that object or *public*. Private code or data is known to and accessible only by another part of the object. That is, private code or data may not be accessed by a piece of the program that exists outside the object. When code or data is public, other parts of your program may access it even though it is defined within an object. Typically, the public parts of an object are used to provide a controlled interface to the private elements of the object. For all intents and purposes, an object is a variable of a user-defined type. It may seem strange that an object that links both

code and data can be thought of as a variable. However, in object-oriented programming, this is precisely the case. Each time you define a new type of object, you are creating a new data type. Each specific instance of this data type is a compound variable.

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iii-object-oriented-programming-with-c-10cs36-notes.pdf