http://www.learncpp.com/cpp-tutorial/116-adding-changing-and-hiding-members-in-a-derived-class/:

**Adding New Functionality:**

Adding new functionality allows you to change code by deriving your own class to meet the need for the new functionality. The point of this is to allow changes to code where they are needed. Also it allows you to access other information in other classes case in point the example from the site:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | class Derived: public Base  {  public:      Derived(int nValue)          :Base(nValue)      {      }        int GetValue() { return m\_nValue; }  }; |

This allows the program to go back up into the base class to look for information if the required information does not exist in the derived class.

**Redefining Functionality:**

When the function is called and it is a derived class function the compiler checks if what is being called is not located in the derived class it then goes up the chain of command until it finds what it is looking for. The problem with this is that if the function you type in doesn’t exist is i.e. Derived::Identify() from the example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int main()  {      Base cBase(5);      cBase.Identify();        Derived cDerived(7);      cDerived.Identify()        return 0;  } |

It goes back to the base and uses the identify function in the base making the function output the statement “I am a Base”. This is not necessarily a bad thing, but can screw up your program if you are not careful. If you want to make sure this doesn’t just write the correct information in the derived class. Though be very very careful when you do that because if you screw up the parameters it can cause all sorts of trouble.

**Adding to Existing Functionality:**

It is possible to add new functions to the class without having to change the entire base class function. All you need to do is call the base function from the derive function just don’t hide the base function instead reuse the code and/or add more functionality to it. An example from the site:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class Derived: public Base  {  public:      Derived(int nValue)          :Base(nValue)      {      }        int GetValue() { return m\_nValue; }        void Identify()      {          Base::Identify(); // call Base::Identify() first          cout << "I am a Derived"; // then identify ourselves      }  }; |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int main()  {      Base cBase(5);      cBase.Identify();        Derived cDerived(7);      cDerived.Identify()        return 0;  } |

When the function executes, it will print out “I am a Base”, “I am a Base”, and “I am a Derived”. The reason for this is because when it goes through the derived function it calls the base::identify() which goes up to the base class and executes the identify in that function. However big warning don’t forget the :: if you do when you call the function it will use the one in the derived class which will trap it into an infinite loop.

**Hiding Functionality:**

When programming in C++ it is possible to hide functionality when you don’t people to mess with it. To do this when you call a public function if the access specifier in the derived class says it is private it will become private. Of course there is another way to keep the public from messing with the function without changing the member. You just have to name the member with the scope resolution operator that allows its access changed in the derived class. This is known as the protected access specifier which keeps the public out but allows the base and derive classes.

http://gamedevelopment.tutsplus.com/tutorials/quick-tip-intro-to-object-oriented-programming-for-game-development--gamedev-1805

**Quick Tip: Intro to Object-Oriented Programming for Game Development Summary:**

The principle of object oriented code is to write clean, modifiable, and executable code that can be reused over and over without the need to copy and paste. It will also help save time and keep people organized with their code when they combine their code with someone else’s code. This principle is important to me because the need to know how to write code in such a matter well greatly help with readability and because I intend to go into the game developing industry so to know this will greatly help my coding skills.

# Quick Tip: The OOP Principle of Cohesion Summary:

# The principle of cohesion is to write a piece of code that only does one thing and it does that one thing well, and that piece of code has to be able to combine with another piece of code to do a task. The point of this is it makes code easier to modify and to keep it focused on what it needs to do. This is very important to me because it helps me stay focused on what that section of code should do and what sections of code it should work with. It will also be a great help with readability because all the object does is one thing you won’t have to scroll through the object and worry about getting lost.

# Quick Tip: The OOP Principle of Coupling Summary:

# The principle of coupling is to lower the direct effect of objects on states and behaviors. I found this to be important because knowing how to change a section of code without it going all over the place and rearranging everything is great to know. I try to practice this already mainly because I don’t want to have to go back debug for eight hours, but sometimes it is hard to avoid.

# Quick Tip: The OOP Principle of Encapsulation Summary:

# The principle of encapsulation is to hide information that objects shouldn’t be allowed to use. It also helps create loose coupling by limiting the access to an object’s states and behaviors. I find this to be very important because when I’m developing programs it helps to know that certain information will not be accessed which can potentially lower the number of errors I have to deal with.

# Quick Tip: The OOP Principle of Abstraction Summary:

# The principle of abstraction is to help you avoid using the same code over and over. This principle is important to me because it will help me use less code to do the same job. It will also help save space and time working on the programming because believe it or not you can screw up copying and pasting a section of code, but this usually occurs when you paste in the wrong area.

# Quick Tip: The OOP Principle of Inheritance Summary:

# The principle of inheritance gives us the ability to make similar objects without having to write code in between them. It also allows us to keep our code more maintainable. I find this to be important because without you can’t develop a hierarchy which would be bad because then you have write the same code in between leading more chances of errors occurring and making the program harder to read.

# http://www.youtube.com/watch?v=K0mgc3efx-A summary:

# I think version control is important because it allows you to see what changes you or someone else made to your code. If an error occurs you know where to begin your search for the error. This would have shorten the project time for fightopia because the groups could have kept in contact whenever they had free time.