Essay #2: Classes, Objects, & Methods: Feb. 22nd, 2016:

C++ Programming

The main purpose of C++ programming is to add object orientation to the C programming language and classes are the central feature of C++ that supports object-oriented programming and are often called user-defined types.

A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.

How To Define A Class

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations. For example, we defined the Box data type using the keyword class as follows:

Example #1:

class Box

{

public:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box

};

What does the keyword “public” mean?

The keyword public determines the access attributes of the members of the class that follow it. A public member can be accessed from outside the class anywhere within the scope of the class object. You can also specify the members of a class as private or protected which we will discuss in a sub-section. A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types.

Data Members of Objects & Classes

The public data members of objects of a class can be accessed using the direct member access operator (.).

|  |  |
| --- | --- |
| Concept: | Description: |
| [Class member functions](http://www.tutorialspoint.com/cplusplus/cpp_class_member_functions.htm): | A member function of a class is a function that has its definition or its prototype within the class definition like any other variable. |
| [Class access modifiers](http://www.tutorialspoint.com/cplusplus/cpp_class_access_modifiers.htm): | A class member can be defined as public, private or protected. By default members would be assumed as private. |
| [Constructor & destructor](http://www.tutorialspoint.com/cplusplus/cpp_constructor_destructor.htm): | A class constructor is a special function in a class that is called when a new object of the class is created. A destructor is also a special function which is called when created object is deleted. |
| [C++ copy constructor](http://www.tutorialspoint.com/cplusplus/cpp_copy_constructor.htm): | The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. |
| [C++ friend functions](http://www.tutorialspoint.com/cplusplus/cpp_friend_functions.htm): | A friend function is permitted full access to private and protected members of a class. |
| [C++ inline functions](http://www.tutorialspoint.com/cplusplus/cpp_inline_functions.htm): | With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function. |
| [The this pointer in C++](http://www.tutorialspoint.com/cplusplus/cpp_this_pointer.htm): | Every object has a special pointer this which points to the object itself. |
| [Pointer to C++ classes](http://www.tutorialspoint.com/cplusplus/cpp_pointer_to_class.htm): | A pointer to a class is done exactly the same way a pointer to a structure is. In fact a class is really just a structure with functions in it. |
| [Static members of a class](http://www.tutorialspoint.com/cplusplus/cpp_static_members.htm): | Both data members and function members of a class can be declared as static. |

Objects & Object Variables:

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Example #2 Of A Basic Class:

class MyClass:

variable = "blah"

def. function (self):

print "This is a message inside the class.

First, to assign the above class (template) to an object you would do the following:

myobjectx = MyClass()

Now the variable "myobjectx" holds an object of the class "MyClass" that contains the variable and the function defined within the class called "MyClass".

Accessing Object Variables

You can create multiple different objects that are of the same class(have the same variables and functions defined). However, each object contains independent copies of the variables defined in the class. For instance, if we were to define another object with the "MyClass" class and then change the string in the variable above:

myobjecty = MyClass()

myobjecty.variable = "yackity"

Then you have to print out both values:

print myobjectx.variable # This would print "blah".

print myobjecty.variable # This would print "yackity".

Accessing Object Functions

To access a function inside of an object you use notation similar to accessing a variable:

myobjectx.function()

Then the above would print out the message, "This is a message inside the class."

Methods: Different Kinds of Methods:

## In the world of computer programming, there are a lot of very different methods. There are object class methods, interface methods, overloading/overriding methods, which include subclasses, each defining the appropriate formula of calculating the area. There are also accessor, mutator and manager methods. The first groups of methods being accessor methods are used for the reading of data values of an object. The second group of methods is mutator methods, which modify the data of an object. The third group of methods is manager methods, those of which initialize and destroy objects on a class, which include constructors and destructors. Notice that a constructor is a method being called at the beginning of an object’s lifetime to create and initialize the object, a process called construction and that a destructor is a method that is called automatically at the end of an object's lifetime, a process called destruction. There are also abstract methods, which have only a signature without an implementation body and often used to specify that a subclass must provide an implementation of the method; class methods, which are called on a class rather than an instance and are typically used as part of an object meta-model; static methods, which are meant to be relevant to all the instances of a class rather than to any specific instance; operator methods, which  define or redefine operator symbols and define the operations to be performed with the symbol and the associated method parameters; and virtual methods, which  are the means by which a C++ class can achieve polymorphic behavior. The only methods *not* participating in polymorphic behaviors are non-virtual aka regular methods.