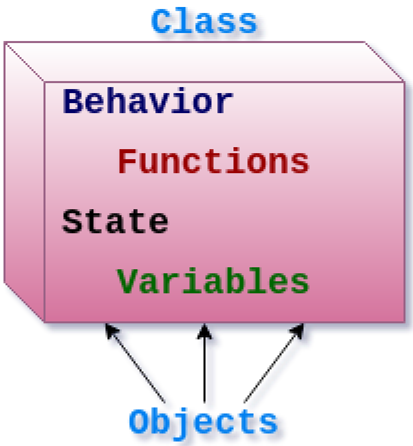
Classes

General Overview



A **class** is similar to a **struct** with many added benefits and features. You can think of a class as a container of information. This allows for multiple pieces of data within one entity. To turn a struct into a class, just add member functions (you would also need to change the keyword from **struct** to **class**). Here are the differences in detail:

A **struct** contains only data types (**char**, **int**, **struct**), does not contain any member functions, and is always **public**.

A **class** contains both data types and member functions and has both **public** and **private** sections and sometimes a **protected** section.

In order to decide if you need a class, consider if the topic at hand is a **noun**. This gives you the ability to have descriptors. For example, if you have a grocery store and want organization storing data, you may make a **class** called produce. Inside this class can contain information such as price, color, name, plu, tax price, if the produce contains seeds, if it is a fruit or vegetable, how to ring up the produce either by weight or quantity, etc.

To keep things simplified, the following examples will contain minimal member functions/variables to be able to express each topic in a terse yet effective manner.

Here is a general shell of a class:

```
class ClassName //ClassName usually uppercase
{
    public:
        //Area that will be visible to outside functions via objects
    private:
       //Confined area to the class/Only visible via public functions
}; //Semicolon to state end of class interface as class is a statement
```

To begin, the keyword **class** followed by the name of the class is put on a line by itself. Underneath is the body of the class which will house the interface with the following sections. By default, a class is **private**. However, standard convention has the first portion of a class as **public**. Therefore, it is necessary to start with the keyword **public**. It is legal to have these sections in any order in a class, and you are able to have multiple sections of the same type (such as two public sections), but this is rarely seen. Remember that **public**, **private**, and **protected** (discussed later) all end with a colon.

Objects Of The Class

Since structs have been discussed, the way to declare an object/instance of a class is the same method. Just like an **int** or **char**, start out by putting the type (class name) followed by a 'variable' (known as an object of the class). Different declaration methods are discussed in the next section. Objects are uppercase.

A Produce class will be created to show the beginning aspects of a class (member variables and constructors).

```
int main()
{   //Type   Object   Object
    Produce Banana, RedPepper; //Basic declaration of 2 objects
}
```

Inside The Class

Keep in mind that a class is an Abstract Data Type. Therefore, the user will never see the implementation of the member functions as well as know how the member variables are used.

The main aspect for the user is to know how to use the constructors, destructor, member functions, and overloaded operators. In order for the user to utilize the aforementioned, they need to be in the **public** section of the class. This is the gateway to the class from outside functions via a class object. The member variables will be placed in the **private** section to ensure they are only manipulated from within the class.

To begin, member variables, or attributes, are added in the **private** section. This is the data of the class that describes the object at hand and also describes the object's current state. These are able to be changed only via member/helper functions.

```
class Produce
{
    Public:
    Private:
        //Member variables
        string name;
        int plu;
        double price;
};
```

Member variables are able to be any data type. One exception to this is that you cannot have an object of the **same** class as that would cause infinite declarations. **Class functions/variables must not be named the same. Ex) You cannot have a function plu() and an int plu.**