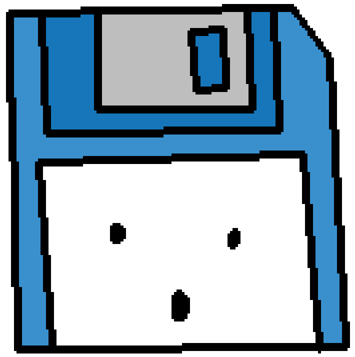
Classes
Part 1

Written by Rachel J. Morris, last updated 2015-01-14

Classes

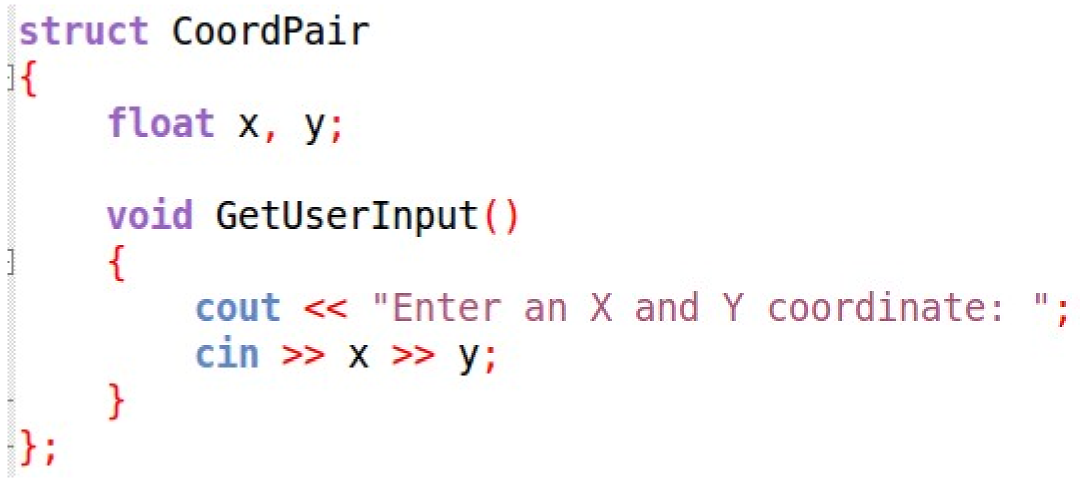
Classes are similar to structs, but have more sophisticated functionality, if you choose to use it.

You can write a class that is exactly the same as a struct, so for simple data, it's up to you which one you want to use.

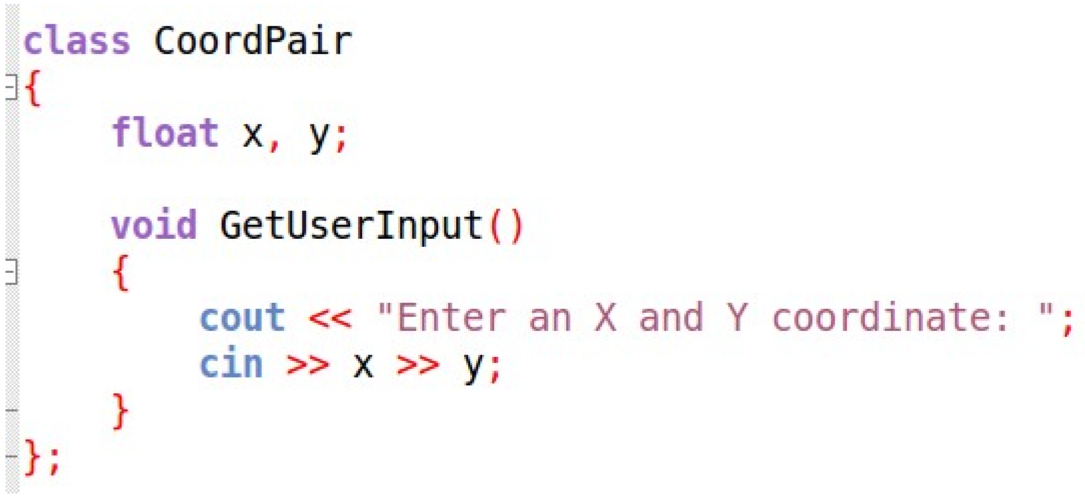


Classes

Declaring a class looks similar to a struct...



```
struct CoordPair
{
    float x, y;

    void GetUserInput()
    {
        cout << "Enter an X and Y coordinate: ";
        cin >> x >> y;
    }
};
```



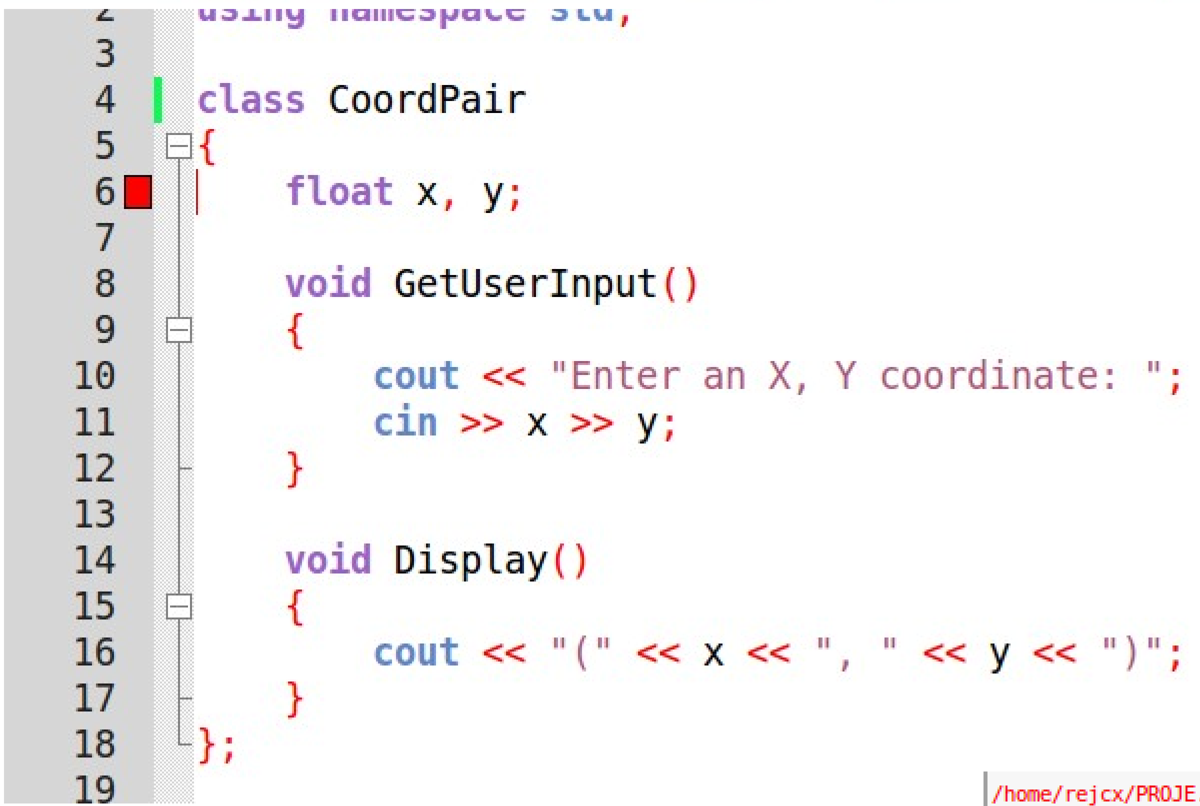
```
class CoordPair
{
    float x, y;

    void GetUserInput()
    {
        cout << "Enter an X and Y coordinate: ";
        cin >> x >> y;
    }
};
```

However...

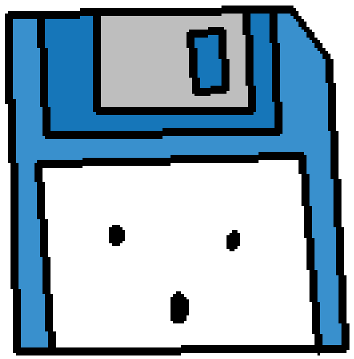
Classes

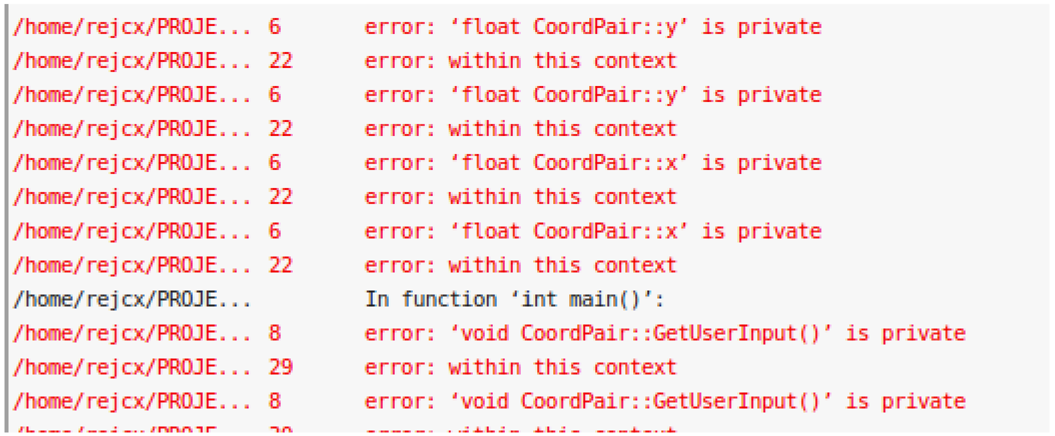
Make a small program with a struct.
Then change it to a class.
Build errors due to privacy.

Classes



```
 2      using namespace std;
 3
 4      class CoordPair
 5      {
 6          float x, y;
 7
 8          void GetUserInput()
 9          {
10              cout << "Enter an X, Y coordinate: ";
11              cin >> x >> y;
12          }
13
14          void Display()
15          {
16              cout << "(" << x << ", " << y << ")";
17          }
18      };
19
```

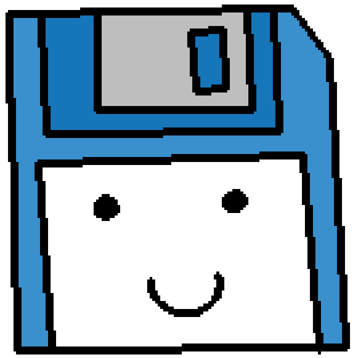




```
/home/rejcx/PROJE... 6      error: 'float CoordPair::y' is private
/home/rejcx/PROJE... 22     error: within this context
/home/rejcx/PROJE... 6      error: 'float CoordPair::y' is private
/home/rejcx/PROJE... 22     error: within this context
/home/rejcx/PROJE... 6      error: 'float CoordPair::x' is private
/home/rejcx/PROJE... 22     error: within this context
/home/rejcx/PROJE... 6      error: 'float CoordPair::x' is private
/home/rejcx/PROJE... 22     error: within this context
/home/rejcx/PROJE...        In function 'int main()':
/home/rejcx/PROJE... 8      error: 'void CoordPair::GetUserInput()' is private
/home/rejcx/PROJE... 29     error: within this context
/home/rejcx/PROJE... 8      error: 'void CoordPair::GetUserInput()' is private
/home/rejcx/PROJE... 29     error: within this context
```

What happened when we changed “struct” to “class”?

Accessibility



Private and Public Members

By default, structs have **public** member variables and functions.

For classes, **private** member variables and functions are the default.

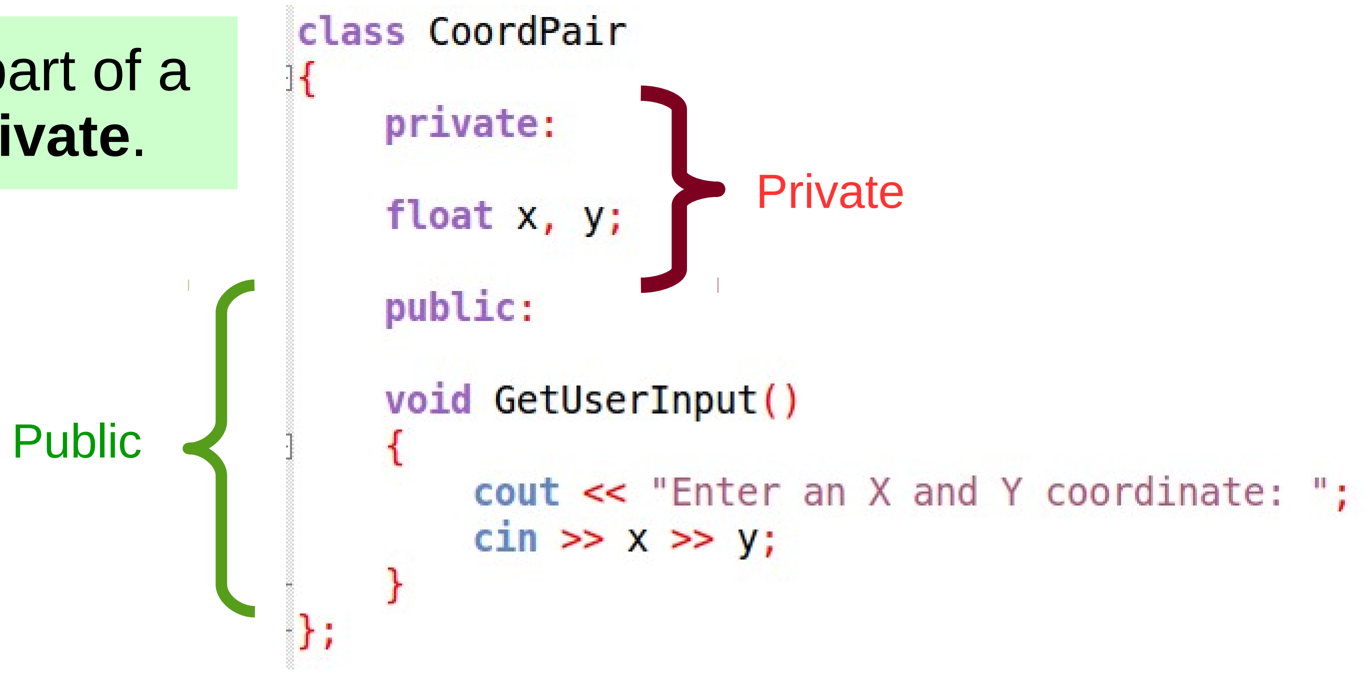
Any **private** variables or functions are **not accessible outside of the class – only from within the class itself.**

Private and Public Members

We can take advantage of **public** and **private** members by hiding the inner-workings of an object, and only displaying functionality that the user (or another programmer) would need.

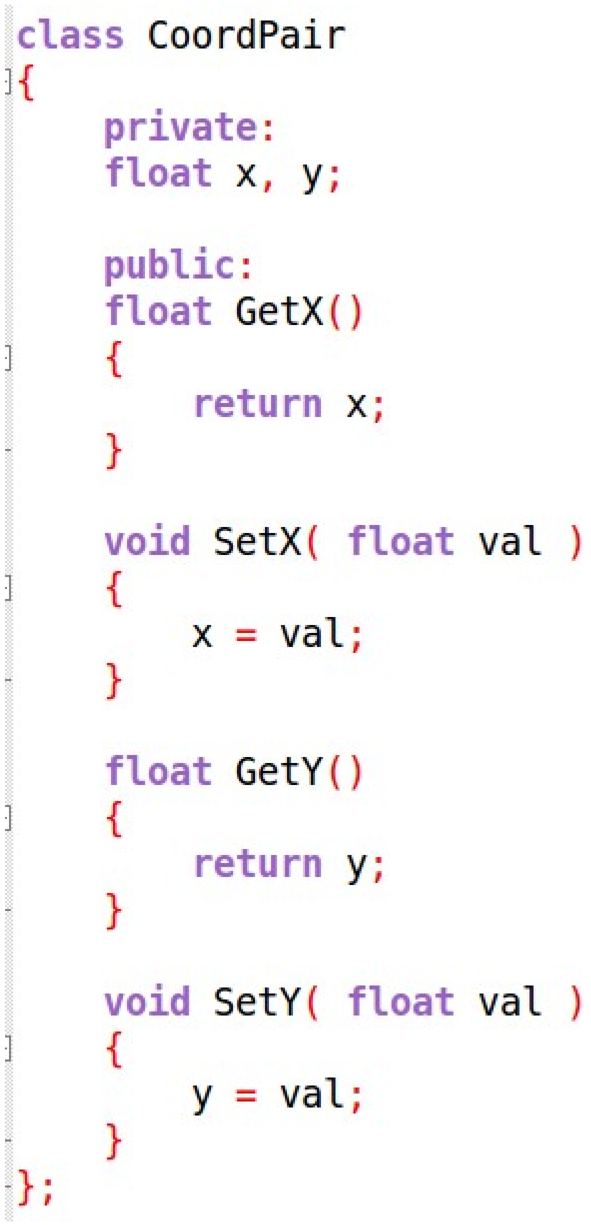
More on the “whys” in a lecture on design...

We can explicitly what part of a class is **public** and **private**.



```
class CoordPair
{
    private:

    float x, y;

    public:

    void GetUserInput()
    {
        cout << "Enter an X and Y coordinate: ";
        cin >> x >> y;
    }
};
```

Getters and Setters



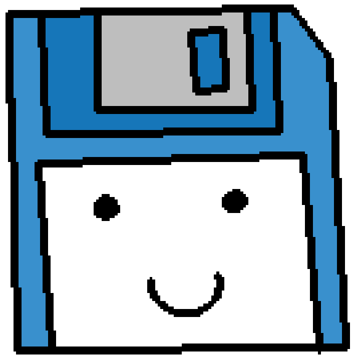
```
class CoordPair
{
    private:
    float x, y;

    public:
    float GetX()
    {
        return x;
    }

    void SetX( float val )
    {
        x = val;
    }

    float GetY()
    {
        return y;
    }

    void SetY( float val )
    {
        y = val;
    }
};
```

It is generally a good idea to make member variables private, and only make them accessible through a function "in-between".

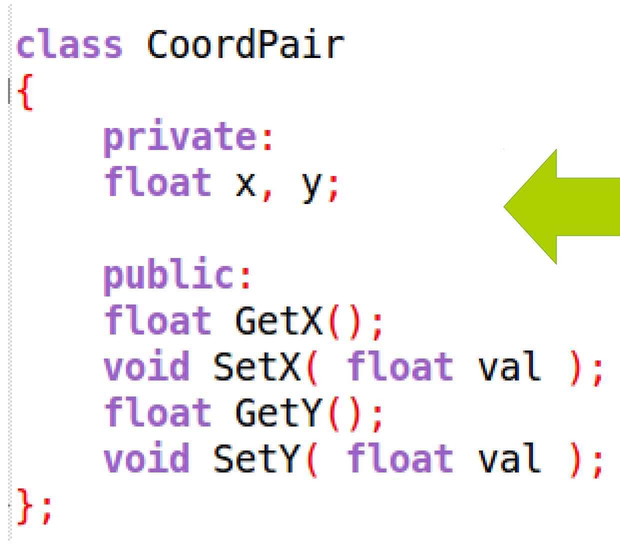
These are generally called **Getters and Setters** Or **Accessor and Mutator** functions.

Keep this in mind for later.

Defining Member Functions



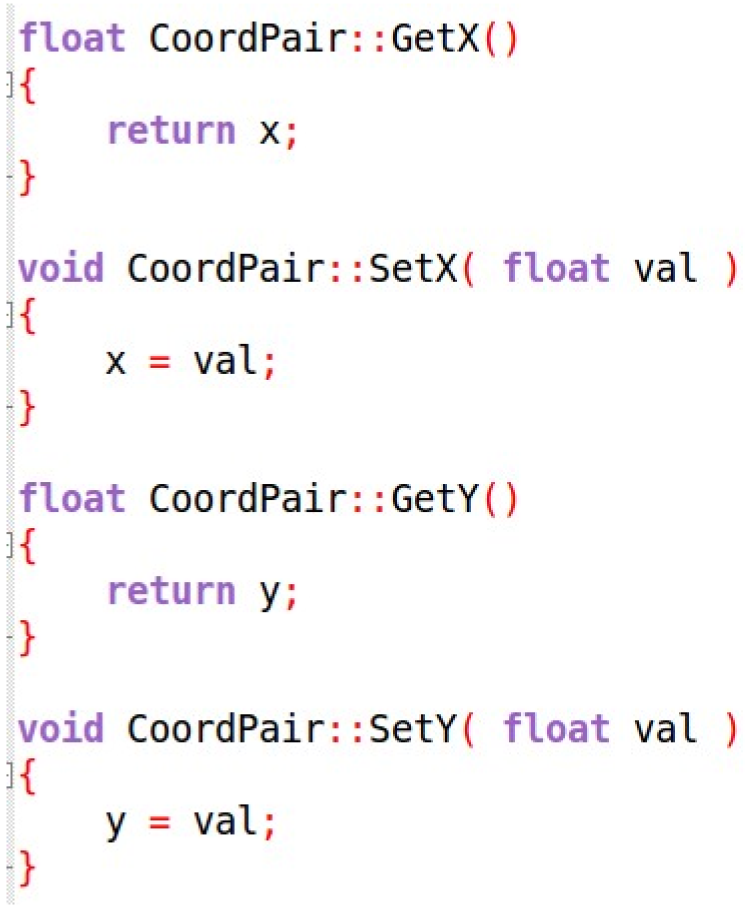
Defining Functions



```cpp
class CoordPair
{
    private:
    float x, y;

    public:
    float GetX();
    void SetX( float val );
    float GetY();
    void SetY( float val );
};
```

While you *can* define your member functions from within the class,
It is generally more standard in C++ to only **declare** functions within the class.

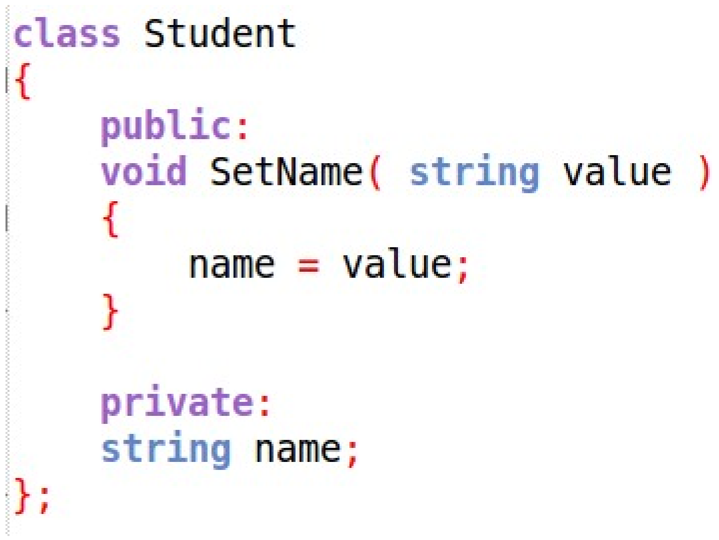
Then we can **define** the member functions outside of the class...



```cpp
float CoordPair::GetX()
{
    return x;
}

void CoordPair::SetX( float val )
{
    x = val;
}

float CoordPair::GetY()
{
    return y;
}

void CoordPair::SetY( float val )
{
    y = val;
}
```

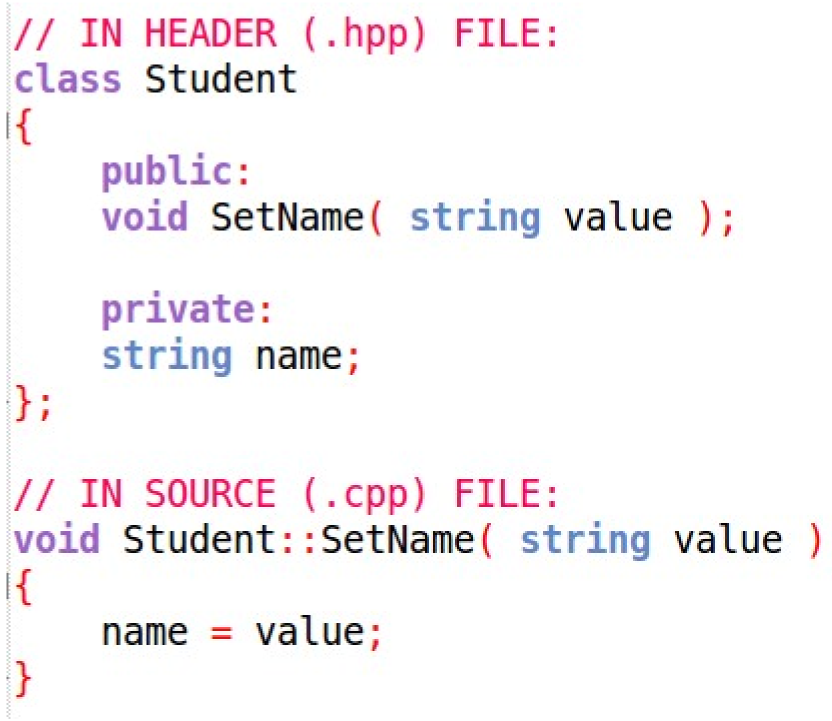
Defining Functions

Defining everything within the class



```cpp
class Student
{
    public:
    void SetName( string value )
    {
        name = value;
    }

    private:
    string name;
};
```

Defining functions outside



```cpp
// IN HEADER (.hpp) FILE:
class Student
{
    public:
    void SetName( string value );

    private:
    string name;
};

// IN SOURCE (.cpp) FILE:
void Student::SetName( string value )
{
    name = value;
}
```

We do this because, in C++, we store the **declarations** in one type of source file (.hpp), and the **definitions** in another type (.cpp).

More on this in the lecture on **using multiple files.**

Defining Functions

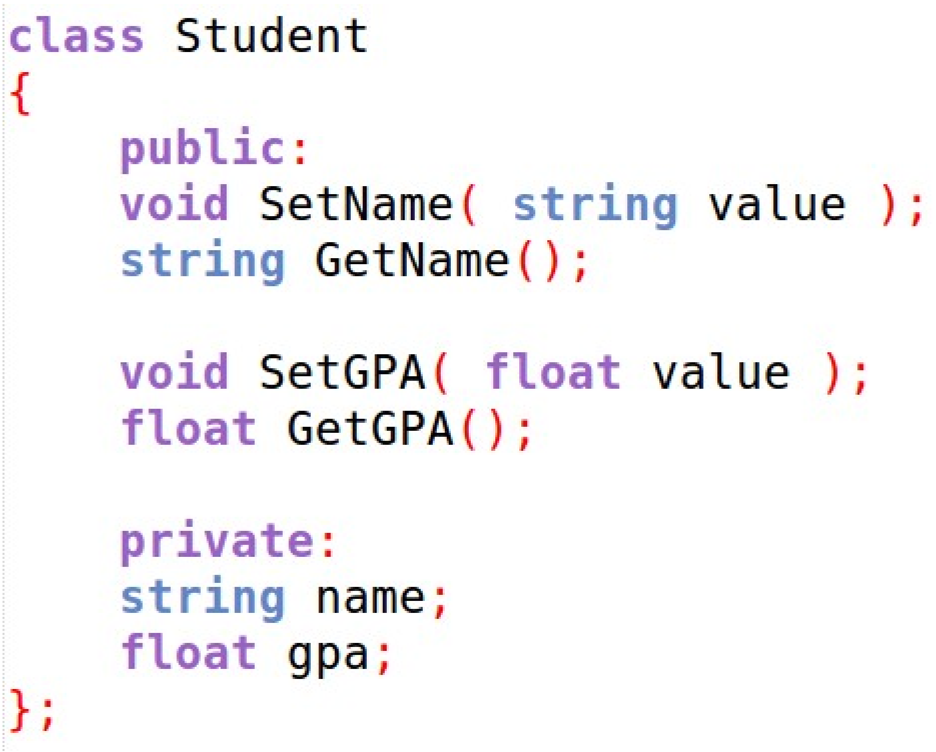
So your class declaration should have
Member variable declarations
And
Member function declarations
But not the function definitions.

Remember that the **definition** is when we have the function body and inner code. The declaration is ***only*** the function header, with a semicolon at the end.

Within the class declaration, functions are not defined.

The user will interface with the Student object via its **public functionality**

Behind-the-scenes code is kept **private**.



```
class Student
{
    public:
    void SetName( string value );
    string GetName();

    void SetGPA( float value );
    float GetGPA();

    private:
    string name;
    float gpa;
};
```

Defining Functions

The function **definitions** are written outside of the class – usually in a separate file.

Remember that the **definition** is when we have the function body and inner code. The declaration is ***only*** the function header, with a semicolon at the end.

```
void Student::SetName( string value )
{
    name = value;
}
```

Notice that when we define the functions in this way, we need to prepend the class name, followed by the **scope resolution operator ::**

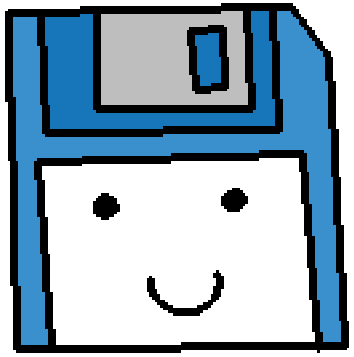
```
string Student::GetName()
{
    return name;
}
```

This is for functions that are members of a class. Not for functions as a whole.

This is only scratching the surface of **classes**.

There's still a lot more to cover, but for now we should do some sample programs!

Classes are very important, so make sure that you practice a lot!



Sample Programs...

Let's write some code to demonstrate how to use classes

A. Chat Log

B. Fractions

C. Game Entity

Sample Programs...

Let's write some code to demonstrate how to use classes

A. Chat Log

B. Fractions

C. Game Entity

Sample Programs...

Let's write some code to demonstrate how to use classes

A. Chat Log

B. Fractions

C. Game Entity

Sample Programs...

Let's write some code to demonstrate how to use classes

A. Chat Log

B. Fractions

C. Game Entity