Class Member Functions

In this lesson, we will highlight the role of member functions in classes.

WE'LL COVER THE FOLLOWING

The Purpose of Member Functions

Member functions act as the interface between a program and the data members of a class in the program.

These functions can either alter the content of the data variables or use their values to perform a certain computation. All the useful member functions should be **public**, although, some functions which do not need to be accessed from the outside could be kept private.

Declaration and Definition

Like all functions, member functions can either be defined straightaway, or they could be declared first and defined later.

Here's an example of a function being defined in the class:

```
class Rectangle {
  int length;
  int width;

  public:
  void setLength(int l){ // This function changes the value of length
    length = l;
  }

  int area(){

    return length * width; // Only the values of the data members are
                           // accessed and used to calculate the area
  }
};
```

Scope Resolution Operator

The scope resolution operator (`::`) allows us to simply declare the member functions in the class and define them elsewhere in the code. To use the scope resolution operator, we follow a certain syntax:

```
returnType className::function()
```

Now, we'll use the scope resolution operator on our `Rectangle` class.

```
class Rectangle {
  int length;
  int width;

  public:

  // We only write the declaration here
  void setLength(int l);
  int area();
};

// Somewhere else in the code
void Rectangle::setLength(int l){ // Using the scope resolution operator
  length = l;
}

int Rectangle::area(){
  return length * width;
}
```

The `::` operator tells the compiler that the particular function belongs to the class preceding it.

Why is this useful? We'll find out the answer in the data hiding section.

Get and Set Functions

These two types of functions are very popular in OOP. A **get** function retrieves the value of a particular data member, whereas a **set** function sets its value.

It is a common convention to write the name of the corresponding member variable with the `get` or `set` command. We have already seen an example of a `set` function in the code above. `setLength(int l)` is a perfect example. Let's write get and set functions for the `length` and `width` in our `Rectangle` class:

```
class Rectangle {
  int length;
  int width;

  public:

  // get and set for length
  void setLength(int l);
  int getLength();

  // get and for width
  void setWidth(int w);
  int getwidth();

  int area();
};


void Rectangle::setLength(int l){
  length = l;
}
int Rectangle::getLength(){
  return length;
}

void Rectangle::setWidth(int w){
  width = w;
}
int Rectangle::getWidth(){
  return width;
}

int Rectangle::area(){
  return length * width;
}
```

Overloading

Member functions can be overloaded just like any other function. This means that multiple member functions can exist with the same name on the same scope, but must have different return types and/or arguments.

At this point, we know all about the attributes of a class. However, the most crucial part is still left to explore. How do we define what happens when an object of our class is created? We'll find out in the next lesson.