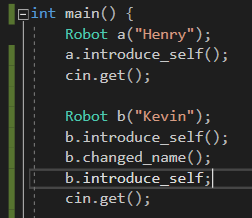
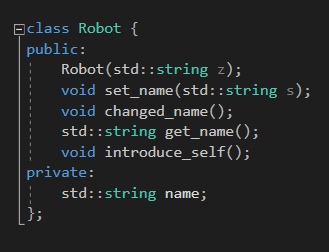
# What is a class?

A class is where the user defines an Overarching group this has a title for the Class, this then can have multiple copies of the same class active at the same time. For example in Mario there can be multiple “Goombas” active at the same time running in tandem.



v

The code shown to the far left is where the class is being defined (in a file called the header) this has to declare the name and type of the ‘methods’ that can be applied to an instance of the class.

You also can set the “Public & Private” status of the properties where the code forcibly locking the access rate of a variable unless you use the classes’ public methods. This adds levels of security to a class as you can use if & case to stop unwanted access or false values to a property.

v

# Explain Encapsulation

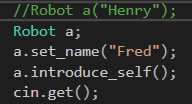
The use of encapsulation was created so that when multiple instances of a class is used the methods and variables used by those objects will be based on the same template. This means the methods which will do the same thing, but each corresponding methods (functions) and properties (variables) are independent of the other objects.

For example if I make a class called Alien and I make an alien called Paul all methods and properties are Paul’s Name & Paul’s Age. Then if I create a different alien called Phil the Name and age would be separate properties than Paul’s correspondent methods.

This is a different approach than procedural programming where for each ‘object’ would need its own variables and methods loose within the code as this would need to return values to different variables. This would become increasingly complex and large in file size.

## How can instances of a class be assigned in main

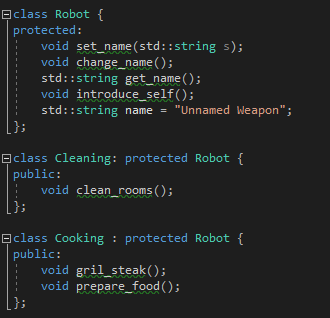
You can either use a Constructor or you can manually create the object by assigning it the data type of the Class you want to use. In this example below I have used both a constructor and without one, when you create an object using a constructor you input something and it uses that give the object properties. Without using a constructor then you can use other methods like set\_name without having to get the user to do it twice.



# Explain Inheritance

Inheritance is a way of creating classes that are getting the methods and properties from another class. A good way of explaining this is that the parent class has some basic properties and methods like get\_age and get\_ colour for a generalised class about Robots). Then you can have specialised classes like Cleaner, Gardener & Chef and Guard. These would have private (so that no new child class could inherit those properties, and that the coder must use the public methods from that class to change private properties. And protected classes allow the user to inherit these properties but they still need to have the classes public methods to edit these properties. Public properties and methods are able to be edited just using the code editor.

Below is an example of how you enable inheritance and sharing of classes’ methods and properties in which you have the original class and below that in the using the colon to show inheritance.



# Explain Polymorphism

Polymorphism is a concept inside OOP where the object’s behaviour and properties to change during different conditions. This inside C++ there are two types of Polymorphism one of them is called Compile Time Polymorphism: which is where the method is overloaded and/or the Operator overloading.

* Method Overloading is where there can be two versions of the same method inside the same class as long as the number of parsed values being different numbers i.e.

Int add(num1, num2) and having another function called Int add(num1, num2, num3) this changes what the function will do so it is Polymorphism.



* Operator Overloading is where the user can add functionality to an Operator so for example you can Overload the operator ‘+’ so that it can concatenate strings as well as add numbers this changes how a function would respond so it is Polymorphism.

Runtime Polymorphism: which is where a parents classes method is overridden by a child class which has declared a method with the same name, when the code is calling a function with the same name it is determined at ‘Runtime’ which is the reason behind the name.

# Evaluate OOP compared to Procedural Programming for Graphical Applications

When using procedural programming you can draw graphical shapes using libraries like turtle and this allows you to draw graphs, these are static images following mathematical formulae. This is useful for applications where the images only need to be drawn and not respond to any stimuli. Procedural Programming is also used inside certain types of Games for example Visual Novels which are when images are shown to the display this is useful as all of these images are shown to the screen using a library, This with a large collection of images will give the impression of movement and just switch images based on user input on top of this would be a series of buttons and keyboard inputs that are set to loop until the next image is shown or one of them is pressed similar to my 16.2. Another way of using procedural programming to show short animations is using the print() command this would allow the user to make symbols move along the screen with repeated statements (like used inside the ZX spectrum) which used ASCII art as well as other Text Types.

When using Object Oriented Programming you have the option of using classes to create editable and responding graphics. This would allow the user to have a class of images that are drawn from files on the computer these then can be edited and changed. For example on the Phaser Game that we created using OOP on JS, for my class Tromps I could then have all of the enemies that are drawn to the screen respond to the displays frame rate. Then using methods that are responsible for enemy movement, health and damage.

OOP also allows the user to class specific instances so when creating a class you do not need individual variables for classes like: Health1, Health2. This would take up time and cause clutter when programming, whereas you would have to do this for procedurally generated groups. When using OOP for games design it allows you to have more easy and simple code to use which improves performance.

For example having a class called ‘Door’ would allow you to have a labelled instance of the class door which handles opening and closing methods, and because its labelled all the doors are completely different instances. Otherwise the code would be long winded or all of the doors would trigger at once.

With OOP for graphical applications you can use inheritance for classes that require similar frameworks and methods and then adapting the methods and properties to be different, for example when you duplicate layers inside a photo editor what the application could do is to create an inherited class from the origin then allow you to change the image to be slightly different: colour, length, hue & saturation, etc.