**Creating Variables and Classes in C++**

Creating variables in C++ is very similar to Java and C#. Here are a few of the most common types used:

int - An integer (Usually 32-bit). Represents whole numbers

float - A floating point number (usually 32-bit). Can represent numbers with decimals

double - A floating point number(usually twice the byte size of a float)

bool - A Boolean. Can represent true or false

char - A character. Represents a singular character (Usually 8-bit)

void - A non-type. Use for functions that do not return a value.

Copy the following code into a project and compile it. This program makes use of all of the types above and initialises them.



Make a note of the difference between an Int, a Float, and a Double. Int’s are initialised by giving it a whole number, Double’s require a decimal point, Float’s also require a decimal point but we add an ‘f’ to the end to distinguish between the two.

Unlike Java, some standard variables like String do not exist as a base type in C++. Luckily, C++ comes with a set of libraries called the Standard Template Libraries which contain templates for objects like Strings, Dynamic Arrays, Maps, etc.

To create a string in C++, you need to make sure the file you are using the string in has the proper include. In this case, we can add the following at the top of the code to allow use to use Strings:



Now all we have to do is define a String and we can use it in our code.



**Operators**

Similar to Java and C#, in C++ there are operators we can use to manipulate variables to produce a new result. The most common operators are below:

* - Subtraction
* + Addition
* \* Multiplication
* / Division
* ++ Increment
* -- Decrement
* = Assignment

There are a lot of others but we will work with the ones above for now. Look into the others though and incorporate them into your exercises. Knowing all the operators is very useful skill.

Below is the code that makes use of all the above operators. A lot of them are very similar if not identical in operation to their counterparts in Java/C#.



Notice above that all the variables are of the type float. It is advisable to keep the types the same but you can add an int and a float together for example as the compiler can cast between these two types.

**Other Useful Formatters**

When working with outputs into the console, there are a couple of extra functions we can use to format the output in a way we want. The two covered here are:

* std::setw(n) – Sets a space of n for between the current and next printed item.
* std::setprecision(n) – Sets the precision of float and double variables that you output.

Both these functions require the iomanip library that you can include at the top of your code files like so:



**A Useful Note**

It can get annoying to keep typing std:: before everything from the standard library or another namespace. We can put the following code at the top of the file and it means we no longer have to prefix all these calls:



**Exercises**

**Creating a Class**

Creating a class in C++ is again very similar to Java and C#. Classes are very good for describing objects in your applications, providing a template with member variables and functions. Below is a class that describes a virtual pet; it knows how hungry a pet is and its name.



Creating a new instance of a class in C++ is identical to Java. The only thing to consider is that the definition of the class (code above) must be before the declaration. The compiler needs to know what the class is before it can create an instance. The following will create a single instance of the CyberPet class:



**Custom Constructors**

Often you will create a class with a custom constructor. You can create as many constructors as you want for a class and the compiler will know which one to choose based on the arguments you give it.



**Exercises**

* Add a Getter and Setter for the name of the pet in the class.
* Create three instances of CyberPet in the main and give each a name and a hunger value.
* Print out the names and hungers of each CyberPet.
* Add a variable that describes how happy the CyberPet is.
* Create a function inside the CyberPet class that prints out the name, hunger and happiness.