C++ NOTE 2:

So we have understood the concepts of classes and objects till now. Also we have gone through the concepts of inheritance and declaring classes and functions.

As per your course we need to understand what Abstract and Concrete classes are. But before we jump into that we need to understand what virtual functions and pure virtual functions are. And also get the basic idea behind pointers and their use in the virtual functions.

We know when we declare a function we give it a specific task to do. But what if that same function has different tasks in two different related classes. Let's say your father has a function doWork() which you have inherited from him. But the meaning of "work" in both of your cases are different. For your father it maybe going to the office and for you it maybe studying, but the function name between you remains same. So we have correctly define which function does what and override the function whenever necessary or whenever the context that is the person is changed. In computer we can do that using a virtual function.

We declare a virtual function with a keyword "virtual". This example should clear up the concept thoroughly. Read the comments.

Eg: https://drive.google.com/file/d/19sPWIgQPKYuLStsMw2LvG3sbeDreK0sC/view?usp=sharing

As you can see in the example that we use pointers to point to the function that we want to call. Pointers are a vast topic that we shall discuss further, but for now you can understand pointers as a way to store addresses of a particular object. So when we say fptr which is a pointer of the father class stores the address of the object, then pointing to a function by that pointer means that the function belongs to that particular object.

So as we understand what virtual functions are now we move towards purely virtual functions. A pure virtual function is a virtual function but with no declaration.

Lets understand this more simply, like when exploring virtual functions we had the doWork() function which had different tasks depending upon who was doing it. But supposedly your father has a function giveExam() which as of now has no meaning to your father cause he is not giving any exams, but you will inherit this function and use this function to attend your boards or any other different exams.

Practically (Read the comments) :
https://drive.google.com/file/d/16iluWB05NpWG5F_9HZDZQKE0nCROmQz9/view?usp=sharing

Definition: A pure virtual function in c++ is a virtual function for which we do not have an implementation. It does not carry any definition related to its base class and is declared by assigning a zero (0) in its declaration.

So now that we understand the concepts of virtual and pure virtual class we move to the real topic that is Abstract and Concrete classes:

Abstract classes: These are classes that contain one or more pure virtual functions which contain no function definition. One thing to note about abstract classes is that you cannot create objects of abstract classes but as you have seen we can create pointers to store the address of the objects of the derived classes.

In the second example, the father class is an abstract class as it gives no definition for the pure virtual function giveExam() but as we see the derived class that is the daughter class gives a

definition to the function. If for some reason the daughter class does not give the definition, it itself becomes an abstract class and the object “Rodosee” cannot be created.

Concrete Classes: As you can guess concrete classes are opposite of Abstract classes, as they contain no pure virtual functions and hence objects can be made from them. Keep in mind Concrete classes can have virtual functions just the pure one’s are barred.

OBJECTS AS FUNCTION ARGUMENTS:

We know that we can pass arguments to a function which are nothing but extra values using which a function performs a certain task. In OOP, we can pass objects of a class as arguments to member functions as well as non-member functions.

There are two ways in which we can pass parameters to functions. They are:

a. Pass by Value: In pass by value we pass the value of arguments to the arguments that are present in the function(Formal arguments) and a copy of these values are stored in different locations than the location of the actual arguments. This ensures that any change occurring in the values does not effect the value in the actual argument. This can also be a major disadvantage for certain systems.

b. Pass by Reference: In pass by reference we create a reference variable for each argument which is passed as function argument and any changes that happen to the reference variable will be also reflected in the actual arguments.

ASSIGNMENT: Find and write the differences between Call by Value and Call by reference and also try to write two programs that show the working of both.