This lecture gives a quick comparison between Cee and cee plus plus.

C plus plus is an object oriented programming language. C is not an object oriented language.

In Cee, we can create a structure as a data type. Imagine a structure called Car.

We can create a Car variable called cee one.

Car is a concept. It has attributes such as brand, model, year, engine size, weight, mileage, etc. We need to create a Car object, such as cee one, then we can talk about a specific car.

A structure is an abstract concept. It does not represent any particular car.

After we create a specific car, then we can talk about this car’s model, year, etc.

We say that an object is an instantiation of a structure.

In the 1970s, as software technologies improved, more people wrote programs. Programs became more complex and mistakes became common.

Researchers studied common mistakes and realized that some common mistakes could be prevented by adding some features to programs.

These features restrict what programmers cannot do for the purpose of preventing mistakes. You can think of these restrictions as lane lines for vehicles on highways. By restricting vehicles into individual lanes, vehicles are less likely to hit each other.

In C, structure and function are not directly connected. It is possible to write new functions processing the same data. The flexibility makes C programs difficult to predict.

In contrast, Cee plus plus links types, called classes, and functions, called methods of the classes. Programmers can restrict only specific functions are allowed to modify objects. These restrictions are enforced by compiler. After a programmer creates a class, it is not possible to create new functions that can modify an object’s attributes. This is called encapsulation.

As an example, suppose we create a class called Person.

We can create a person object and initialize this person’s date of birth. A person’s date of birth should never be changed. This can be enforced by making the attribute for the date of birth a private attribute.

Cee plus plus allows specific ways to create objects. These special functions are called constructors. By using constructors to create objects, these special functions can ensure that all attributes are initialized properly. Every object must be created by the specific constructor. Thus, all attributes are always properly initialized. This is particularly helpful when an object has attributes that require allocating memory.

Cee plus plus also allows destructors to properly free memory. The destructors free all memory and these are the only ways to remove objects. Thus, memory is always properly freed and memory leak can be prevented.

One important concept in object-oriented programming is inheritance. Inheritance is a way to improve code reuse.

Consider this case. We want to create different shapes called square, triangle, and circle. These different types of shapes can share some common attributes, such as the line style and the fill color. Thus, we can create a base class called shape.

Then, we can create derived classes called square, triangle, and circle. The common attributes are handled by the based class. The code for the base class can be reused more easily. The unique attributes are handled by the derived classes.

For the square shape, it has attribute of the length of a side. For the triangle shape, it has attributes of width and height. For the circle shape, it has attribute of radius.

A related concept in object oriented programming is called polymorphism. It means the same function may behave differently depending on which class is referred to.

We want to write a function for calculating the area of a shape. This function cannot be specified at the base class because different types of shares need to calculate the areas differently.

For example, for the square shape, the area is the square of sides.

For the triangle shape, the area is the width times height divvied by two.

For the circle shape, the area is pai times square of the radius.

Cee plus plus has many features and it is not possible to explains all details in a single video.

Here are some commonly misunderstood concept.

First, some people say every valid Cee program is also a valid cee plus plus program. This is wrong. The reason is that cee plus plus has more reserved words. Reserved words cannot be used as the names of variables. It is possible to write a valid cee program that uses a reserved word in cee plus plus but this is not reserved in cee. Thus, it is incorrect to say that every valid Cee program is also a valid cee plus plus program.

Another common mistake is thinking that cee plus plus is slow. Cee plus plus has a concept called copy constructor. It is automatically generated if a programmer does not create one. The problem is that some programmers do not understand the concept of copy constructor and do not understand why copy constructors are invoked. If copy constructors are invoked unintentionally, cee plus plus programs can be slow.

Cee plus plus provides a convenient way to write functions called overloading. Overloading allows different functions with the same names. These functions are selected based on the arguments’ types.

Cee plus plus also allows operators to be overloaded.

Function overloading and operator overloading can be convenient but they are not essential for object oriented programming.

Some people claim that there is no need to use cee plus plus. The additional features are not necessary. It is possible to write every cee plus plus program using cee only. However, some features in cee plus plus can make some programs easier to write. It is also true that you do not need a screw driver. You can always use a hammer to hit a screw. You also do not need scissors because you can always use a knife. You also do not need cars because you can always walk. The point is that some tools are better than the others for certain things.

If anyone tells you these wrong concepts, please quietly and politely walk away. This person does not know cee plus plus. Do not learn object oriented programming from this person.