**Pointer** is a variable in **C++** that holds the address of another variable. They have data type just like variables, for **example** an integer type **pointer** can hold the address of an integer variable and an character type **pointer** can hold the address of char variable.

The variable that stores the address of another variable (like foo in the previous example) is what in **C++** is called a **pointer**. **Pointers** are a very powerful feature of the language that has many uses in lower level programming. A bit later, we will see how to declare and use **pointers**.

A **pointer** is a variable that stores the address of another variable. Unlike other variables that hold values of a certain type, **pointer** holds the address of a variable. For **example**, an integer variable holds (or you can say stores) an integer value, however an integer **pointer** holds the address of a integer variable.

**OOP** stands for **Object-Oriented** Programming. Procedural programming is about writing procedures or functions that perform operations on the data, while **object-oriented** programming is about creating objects that contain both data and functions. ... **OOP** provides a clear structure for the programs.

**C++** provides facility to specify that the compiler should match function calls with the correct definition at the run time; this is called **dynamic binding** or late **binding** or run-time **binding**. **Dynamic binding** is achieved using virtual functions. Base class pointer points to derived class object.

Method Overriding is a perfect **example** of **dynamic binding** as in overriding both parent and child classes have same method and in this case the type of the object determines which method is to be executed. The type of object is determined at the run time so this is known as **dynamic binding**.

In **static binding**, the function defination and the function call are linked during the compile-time whereas in **dynamic binding** the function calls are not resolved until runtime. So they are not bound until runtime. **Static binding** happens when all information needed to call a function is available at the compile-time.

**Class**: A **class** in **C++** is the building block, that leads to Object-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that **class**. A **C++ class** is like a blueprint for an object.

Everything in Java is associated with **classes and objects**, along with its attributes and methods. For **example**: in real life, a car is an **object**. The car has attributes, such as weight and color, and methods, such as drive and brake. A **Class** is like an **object** constructor, or a "blueprint" for creating **objects**.

Everything in C++ is associated with **classes** and objects, along with its attributes and **methods**. For example: in real life, a car is an object. The car has attributes, such as weight and color, and **methods**, such as drive and brake. Attributes and **methods** are basically variables and functions that belongs to the class.