Intro to C++

virtual keyword, polymorphism, vtables

class Pokemon {
    public:
    string name;
    virtual void attack();
    virtual void defend();
}

This class contains two virtual functions
class Pokemon {
    public:
    string name;
    virtual void attack();
    virtual void defend();
};

This class contains two virtual functions

Each class with virtual functions or derived from a class with virtual functions gets its own vtable

Every virtual function has an entry in the vtable which points to the function
When the vtable for the class is created a hidden pointer that points to the vtable is created. Called *_vptr here.

Then when a call is made to a function it follows the *_vptr and find the pointer to the correct function to call.
class Pikachu: Pokemon {
public:
    *__vptr
    void attack();

Pokemon defend function

The most recently derived function is always replaced in that class’ vtable
This ensures that only the most recently derived function will be called when the vtable lookup is done

Pikachu pika;
Pokemon *poke = &pika
poke->defend();  *__vptr->defend_fncptr
poke->attack();   *__vptr->attack_fncptr

Pikachu vtable
attack_fncptr
defend_fncptr