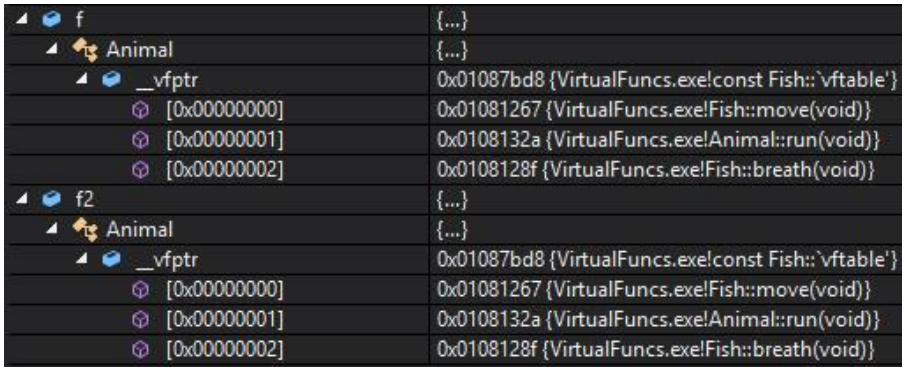
Práctica 15 – Funciones Virtuales

Julián Jiménez González

1. ¿Dónde está situada la tabla de funciones virtuales?

La tabla de funciones virtuales no tiene un lugar fijo, cada compilador puede situarla en un lugar distinto, aunque lo normal es que se haga en el **Data Segment**. Lo que guarda cada instancia de un objeto con funciones virtuales, es un **puntero a esa tabla**.

Como se puede ver a continuación, creando 2 instancias de la clase **Fish**, ambas almacenan un puntero a la tabla, situada en **0x01087bd8**.



2. ¿Cuánto espacio ocupa adicionalmente un objeto por tener una tabla de funciones virtuales?

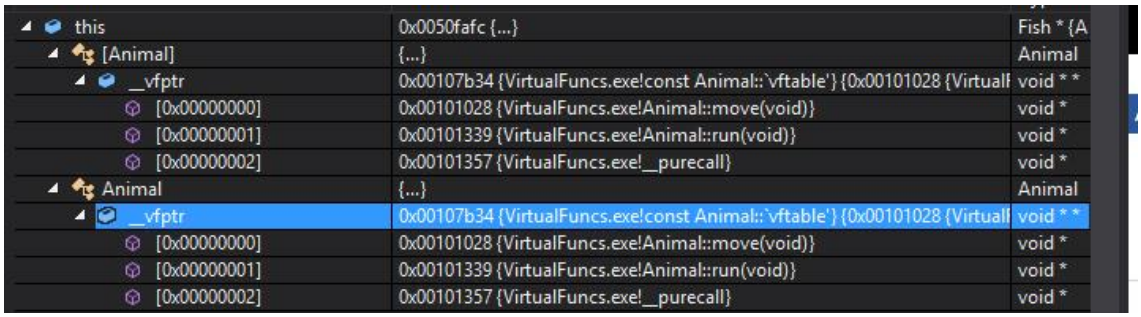
Viendo el mismo ejemplo de antes, el objeto en sí ocupa tan sólo un puntero a mayores (depende de la arquitectura, típicamente 4 u 8 bytes). La tabla sí que ocupa a mayores un puntero por cada función declarada en la clase que sea virtual.

3. ¿Qué pasa si llamo a un método virtual desde el constructor?

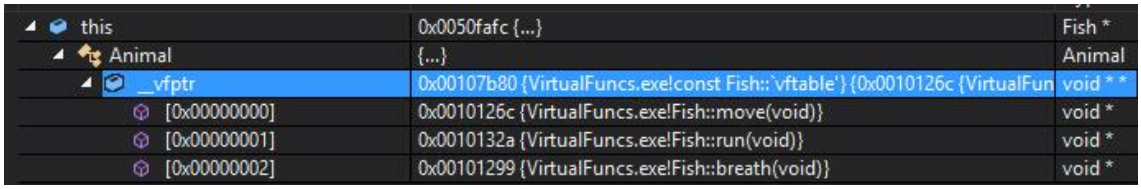
No ocurre nada extraño si el método está definido en la clase base, o en su defecto es un método virtual puro en la base y está definido en la propia clase, y **la llamada se realiza desde el constructor de la derivada**.

Si existe una llamada en el constructor de la clase base, lo que ocurre es que antes de ejecutar el constructor de la derivada se llama al de la base. Como todavía no está creado el objeto, la tabla de funciones virtuales del objeto todavía no está definida, y por defecto se va a usar la de la clase base. En el siguiente ejemplo se puede ver, dadas las clases Fish y Animal, que antes de ejecutar el constructor de Fish su _vfptr apunta a la tabla de Animal.

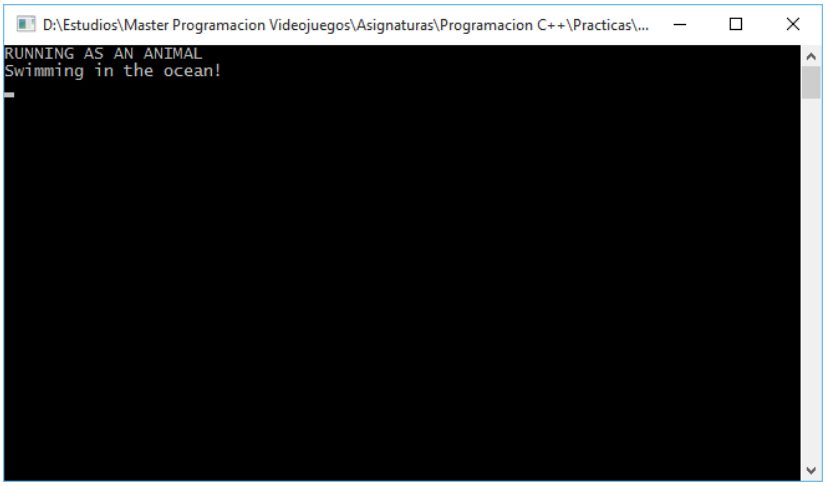
```
Animal() { this->run(); }


Fish::Fish() {
	this->move();
}
```



Una vez que se entra en la ejecución del constructor de Fish, se crea su propia tabla y _vfptr apunta a la nueva dirección.



Por lo tanto, antes de ejecutar el constructor de Fish, se ejecuta el de Animal, que tiene una llamada a run(), y después se ejecuta el propio de Fish, que llama a move(). Este es el resultado:



4. **Comparar la llamada a una función virtual con la llamada a una función no virtual. ¿Cuántos pasos adicionales tienen que realizarse para llamar a una función cuando esta es virtual?**

Para ver la diferencia, se puede añadir el método **jump()** que hay en la clase **Fish**, como un método virtual de **Animal**, y lo podemos reescribir o no en **Horse** (se usa en ambas clases para ver que funciona igual, pero vale con comparar las llamadas a run() y a jump() cuando una es virtual y otra no). Entonces, se puede ver a continuación la cantidad de instrucciones en ensamblador que se ejecutan (**run()** también es un método virtual):

```
pFish->run();
003C1E03  mov         eax,dword ptr [pFish]
003C1E06  mov         edx,dword ptr [eax]
003C1E08  mov         esi,esp
003C1E0A  mov         ecx,dword ptr [pFish]
003C1E0D  mov         eax,dword ptr [edx+4]
003C1E10  call        eax
003C1E12  cmp         esi,esp
003C1E14  call        __RTC_CheckEsp (03C1145h)
      pFish->jump();
003C1E19  mov         eax,dword ptr [pFish]
003C1E1C  mov         edx,dword ptr [eax]
003C1E1E  mov         esi,esp
003C1E20  mov         ecx,dword ptr [pFish]
003C1E23  mov         eax,dword ptr [edx+0Ch]
003C1E26  call        eax
003C1E28  cmp         esi,esp
003C1E2A  call        __RTC_CheckEsp (03C1145h)
      pHorse->jump();
003C1E2F  mov         eax,dword ptr [pHorse]
003C1E32  mov         edx,dword ptr [eax]
003C1E34  mov         esi,esp
003C1E36  mov         ecx,dword ptr [pHorse]
003C1E39  mov         eax,dword ptr [edx+0Ch]
003C1E3C  call        eax
003C1E3E  cmp         esi,esp
003C1E40  call        __RTC_CheckEsp (03C1145h)
```

Si, en cambio, eliminamos el *virtual* del método **jump()**, y ambas clases lo implementan, se puede ver a continuación la cantidad de instrucciones que requieren ambas llamadas para ejecutarlo:

```
      pFish->run();
00841DF3  mov         eax,dword ptr [pFish]
00841DF6  mov         edx,dword ptr [eax]
00841DF8  mov         esi,esp
00841DFA  mov         ecx,dword ptr [pFish]
00841DFD  mov         eax,dword ptr [edx+4]
00841E00  call        eax
00841E02  cmp         esi,esp
00841E04  call        __RTC_CheckEsp (0841145h)
      pFish->jump();
00841E09  mov         ecx,dword ptr [pFish]
00841E0C  call        Fish::jump (08413CAh)
      pHorse->jump();
00841E11  mov         ecx,dword ptr [pHorse]
      pHorse->jump();
00841E14  call        Horse::jump (08413CFh)
```