Object Distinction

程序设计模型

- procedural model

```
char boy[] = "Danny";
char *p_son;
p_son = new char[strlen(boy) + 1];
...
```

- abstract data type model(ADT)

```
String girl = "Anna";
String daughter;

// String::operator=();
daughter = girl;
```

- object-oriented model

```
void check_in(Library *pmat) {
    if(pmat->late())
        pmat->fine();
}
```

- 直接或间接处理继承体系中的一个 base class object， 但只有通过pointer或reference的间接处理，才支持OO程序设计所需的 多态性质。

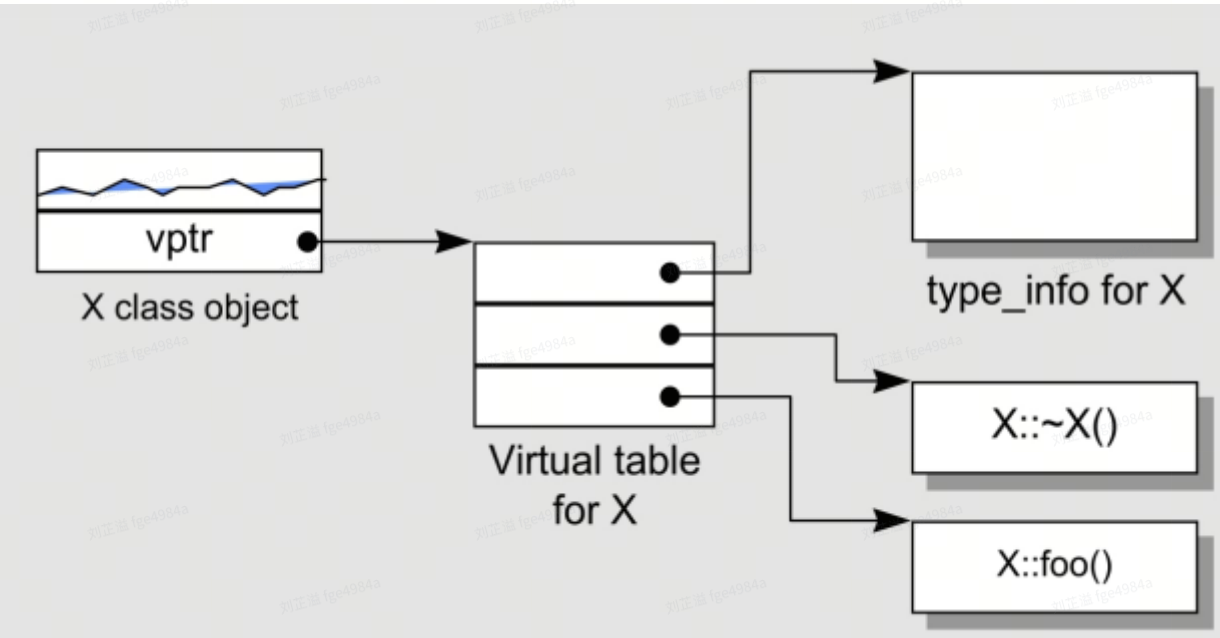
```
Library thing1;
// class Book: public Library{...}
Book book;

// book被sliced(裁剪了);只含有Library相关函数
// 触发Library::check_in()
thing1 = book;

```

```
// thing2指向book
Library &thing2 = book;
// 触发Book::check_in()
thing2.check_in();
```

虚函数表

```
class X {
  private:
    //...
  public:
    //...
    virtual ~X();
    virtual foo();
};
```

- 每一个 class产生出一堆指向 virtual functions 的指针，放在表格 之中。这个表格被称为 **virtual table（vtbl）**
- 每一个 class object被安插一个指针，指向相关的 virtual table。 通常这个指针被称为 **vptr** 。vptr 的设定（setting）和重置（resetting） 都由每一个 **class的 constructor、destructor和 copy assignment运算符**自动完成。
- 每一个 class所关联的 **type_info object**（用以支持 runtime type identification，RTTI）也经由 virtual table被指出来，通常放在表格的第一个 slot。



多态

- 编译器在初始化及指定（assignment）操作（将一个class object指定给另一个class object）之间做出了仲裁。编译器必须确保如果某个object含有**一个或一个以上的vptrs**，那 些vptrs的内容不会被base class object初始化或改变。
- 当一个base class object 被直接初始化为（或是被指定为）一个derived class object时，derived object就会被**切割（sliced）**以塞入较小的base type内存中
- 多态是一种威力强大的设计机制，允许你继一个抽象 的public接口之后，封装相关的类型。需要付出的代价就是额外的间接性——不论是在“内存的获得”或是在“类型的决断”上。

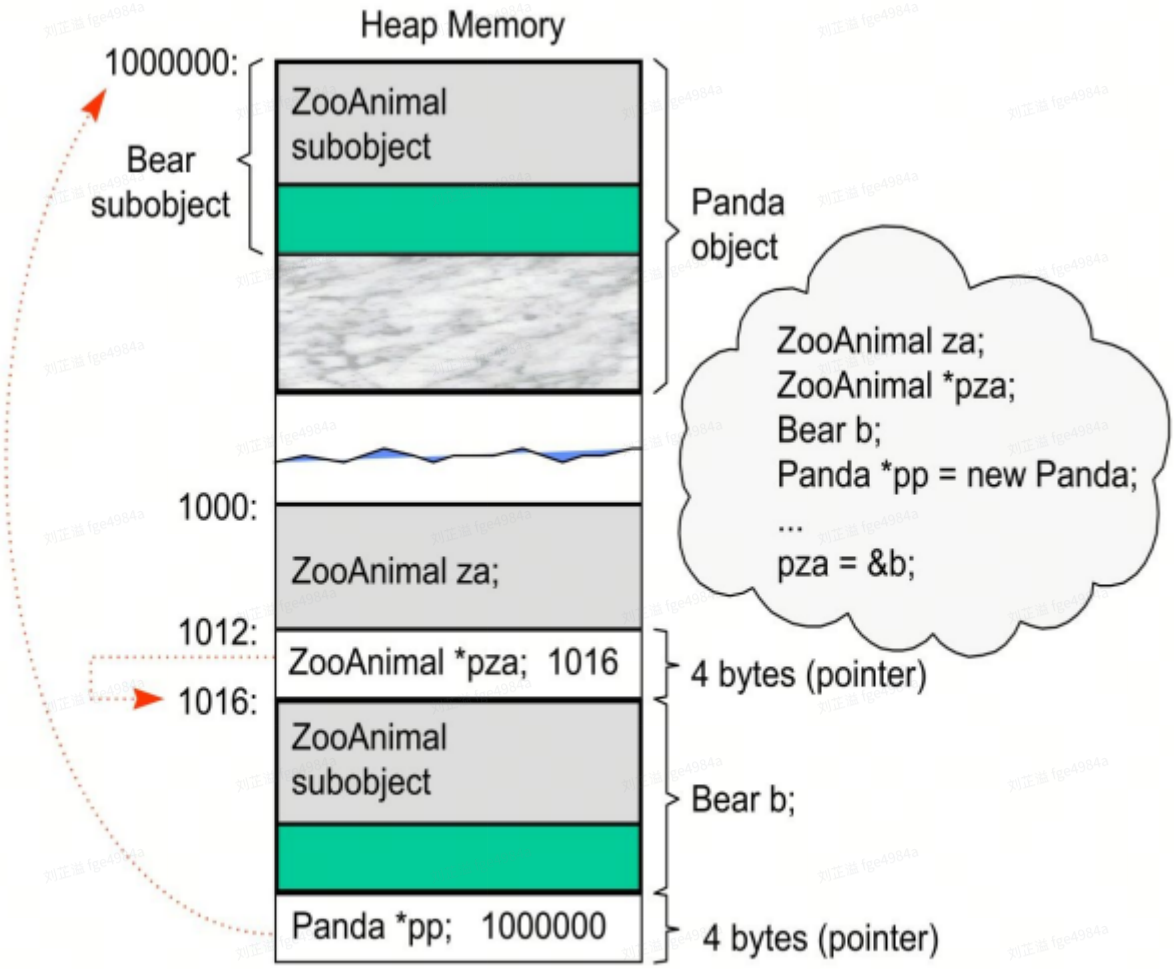


图1.6 依序定义下的内存布局