深度探索C++对象模型

Data语意学

```
class X();
class Y:public virtual X{};
class Z:public virtual X{};
class A:public Y,public Z{};

sizeof X的结果是1
sizeof Y的结果是8(大小和机器，编译器有关)
sizeof Z的结果是8(同上)
sizeof A的结果是12
```

Y和Z的大小受到三个因素的影响：

1. 语言本身造成的额外负担
2. 编译器对于特殊情况所提供的优化处理
3. Alignment的限制

一个virtual base class subobject只会在derived class中存在一份实体，不管他在继承体系中出现了多少次，class A的大小由下列几点决定：

- 被大家共享的唯一一个class X实体，大小为1bytes
- base class Y的大小，减去因virtual base class X而配置的大小，结果是4bytes，base class Z的算法亦同，加起来是8 bytes.
- class A自己的大小：0 bytes
- class A的alignment数量（如果有的话），前述三项综合，表示调整前的大小是9 bytes,class A必须调整至4 bytes边界，所以需要填补3 bytes，结果是12 bytes.

class的data members包括：

- nonstatic data members 放置的是个别的class object感兴趣的数据
- static data members 则放置的是整个class感兴趣的数据

每一个class object因此必须有足够的大小以容纳它所有的nonstatic data members,有时候会比想象中的大：

1. 由编译器自动加上的额外的data members，用以支持某些语言特性（主要是各种virtual特性）
2. 因为alignment（边界调整）的需要

data member的布局

nonstatic data member在class object中的排列顺序将和其被声明的顺序一样，任何中间介入的static data member都不会被放进对象布局之中。

C++ standard要求，在同一个access section中，members的排列只需要符合较晚出现的members在class object中有较高的地址这一条件即可。

members的边界调整可能都需要填补一些bytes。

C++也允许编译器将多个access section之中的data members自由排列，不必在乎它们出现在class 声明中的次序。

data member的存取

static data member

被编译器提出于class之外，并被视为一个global变量。

若取一个static data member的地址，会得到一个指向其数据类型的指针，而不是一个指向其class member 的指针，因为static member并不内含在一个class object之中。

如果有两个classes，每一个都声明了一个static member freelist，,那么当他们都被放在程序的data segment时，就会导致名称冲突，编译器的解决办法hi暗中对每一个static data member 编码，以获得独一无二的程序识别代码，每一种编译器的name-mangling做法都不同

做法都不同。

任何name-mangling做法都有两个要点：

1. 一种算法，推导出独一无二的名称。
2. 万一编译系统必须和使用者交流，那些独一无二的名称可以轻易被推导回原来的名称。

nonstatic data members

nonstatic data members直接存放在每一个class object之中，除非经由明确或暗喻的class object，没有办法直接存取他们。

继承与data member

在C++继承模型中，一个derived class object所表现出来的东西，是其自己的members加上其base class members的总和，至于derived class members和base class members的排列次序并未在C++ standard中强制指定：理论上编译器可以自由安排