H10 PB18000290 胡毅翔

1.阅读[C++:94—类继承（菱形继承、虚继承(virtual虚基类)）](https://blog.csdn.net/qq_41453285/article/details/104442356)，请自行编写简单的具有菱形继承的C++程序（B和C从A继承，D从B、C继承），

1）练习用nm和demangle的方 法分析程序中的2个名字的改编；

g++ c.cpp

得到a.out

nm a.out

部分输出：

000000000000091e W \_ZN1A2aaEi

0000000000000958 W \_ZN1B4funcEv

nm --demangle a.out

部分输出：

000000000000091e W A::aa(int)

0000000000000958 W B::func()

分析：

\_Z mangled符号的前缀

N nested names的前缀

1B,2aa等 <length, id>

E 结束标志

v,i等 参数类型

故\_ZN1A2aaEi表示A::aa(int)；\_ZN1B4funcEv表示B::func()。

1. 试分析在虚继承和非虚继承下的对象布局和方法表布局，并给出你的分析方法。

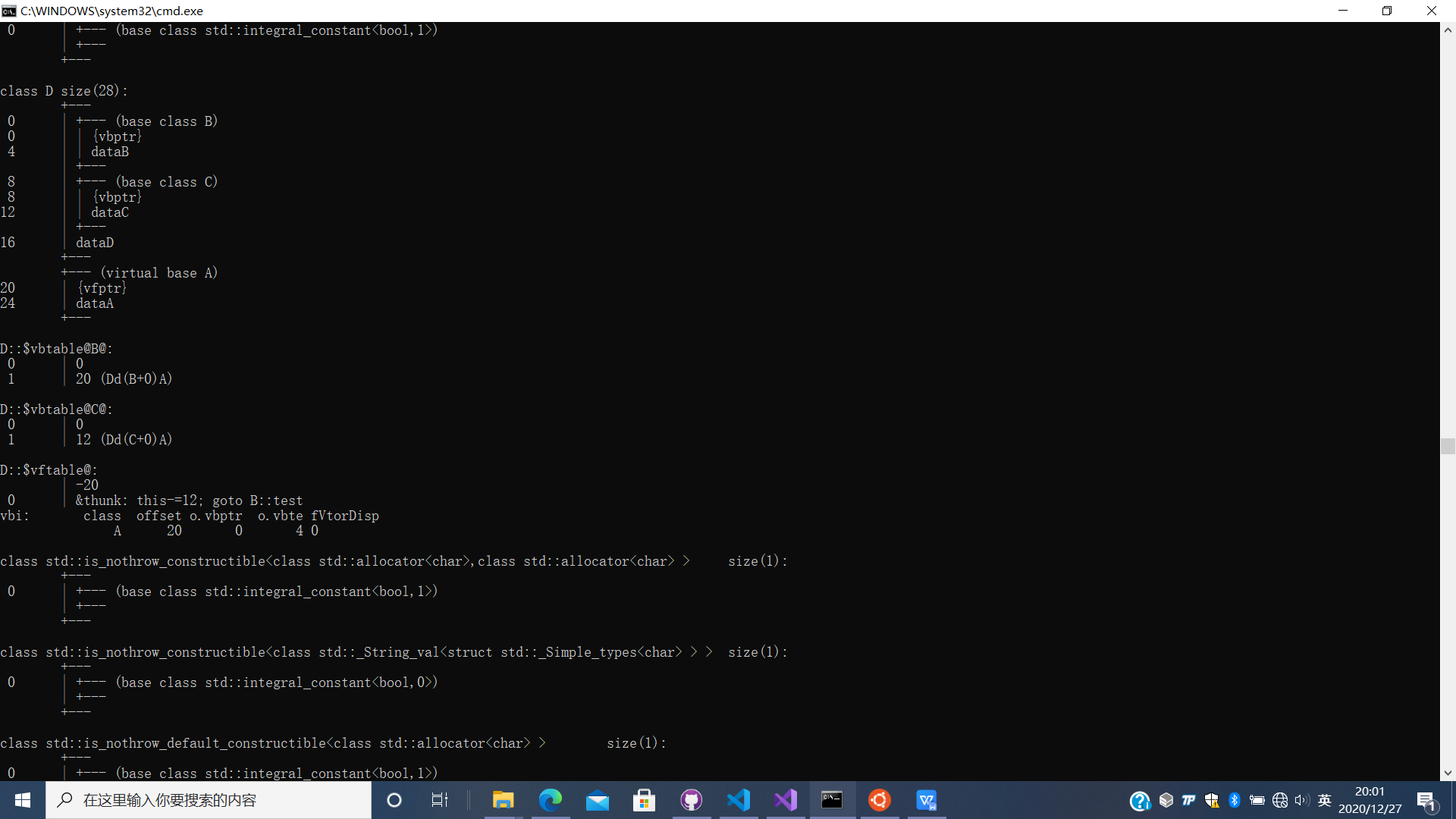
使用Visual Studio中的开发者命令行提示，在c.cpp(非虚继承),c1.cpp(虚继承)所在目录下，输入命令

cl /d1 reportSingleClassLayout[ClassName] [FileName]

即可得到ClassName所对应的类的内存布局，本次实验所得如下：

虚继承：

D的对象布局：



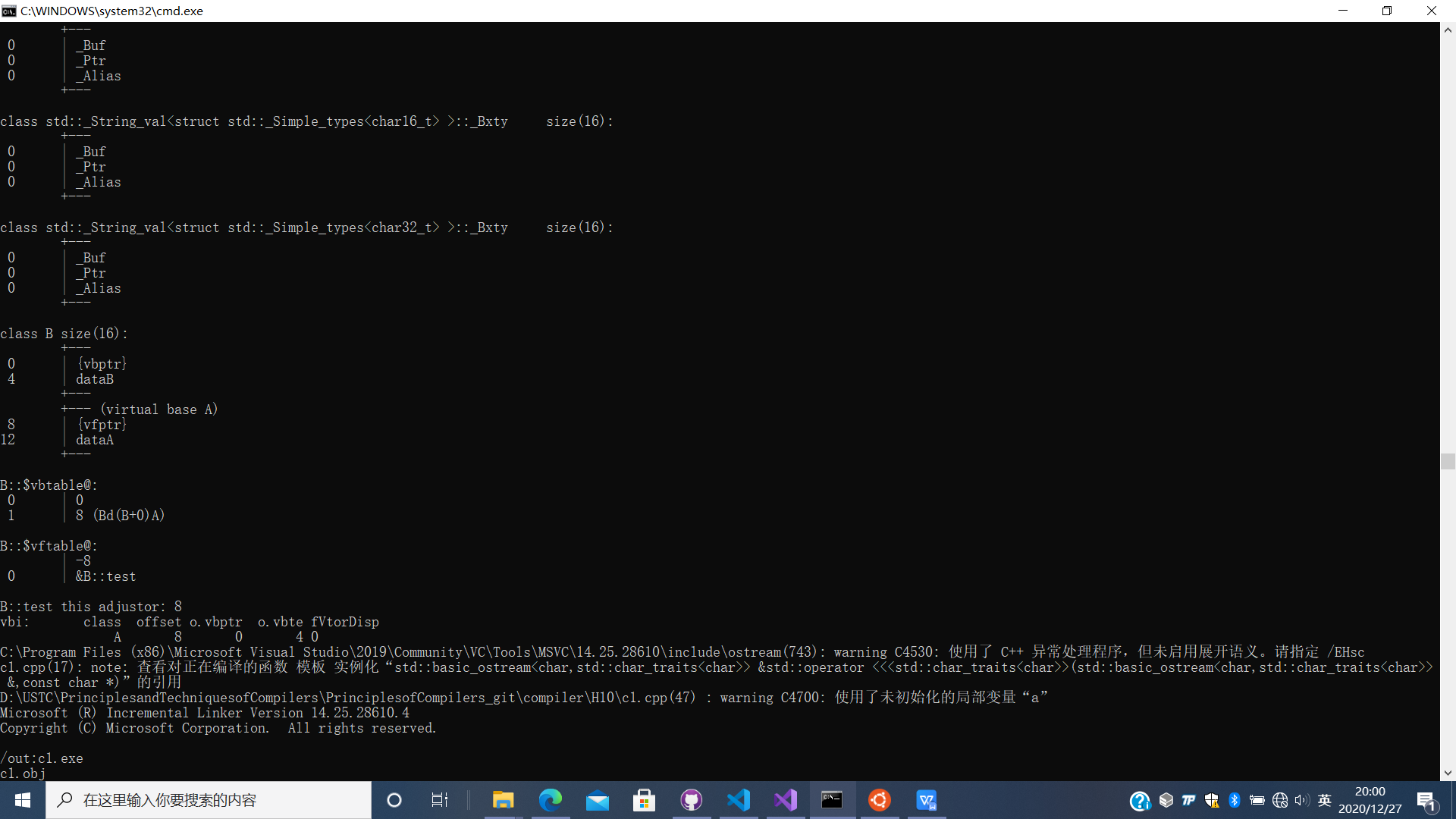
可以看出，虚继承时类D的对象布局与非虚继承时(见后文)有明显不同。

|  |  |  |
| --- | --- | --- |
| 地址(按字节) | 虚继承 | 非虚继承 |
| 0 | vbptr继承自父类B中的指针 | dataA |
| 4 | dataB | dataB |
| 8 | vbptr继承自父类C中的指针 | dataA |
| 12 | dataC | dataC |
| 16 | dataD | dataD |
| 20 | vfptr虚函数表指针 |  |
| 24 | dataA |  |

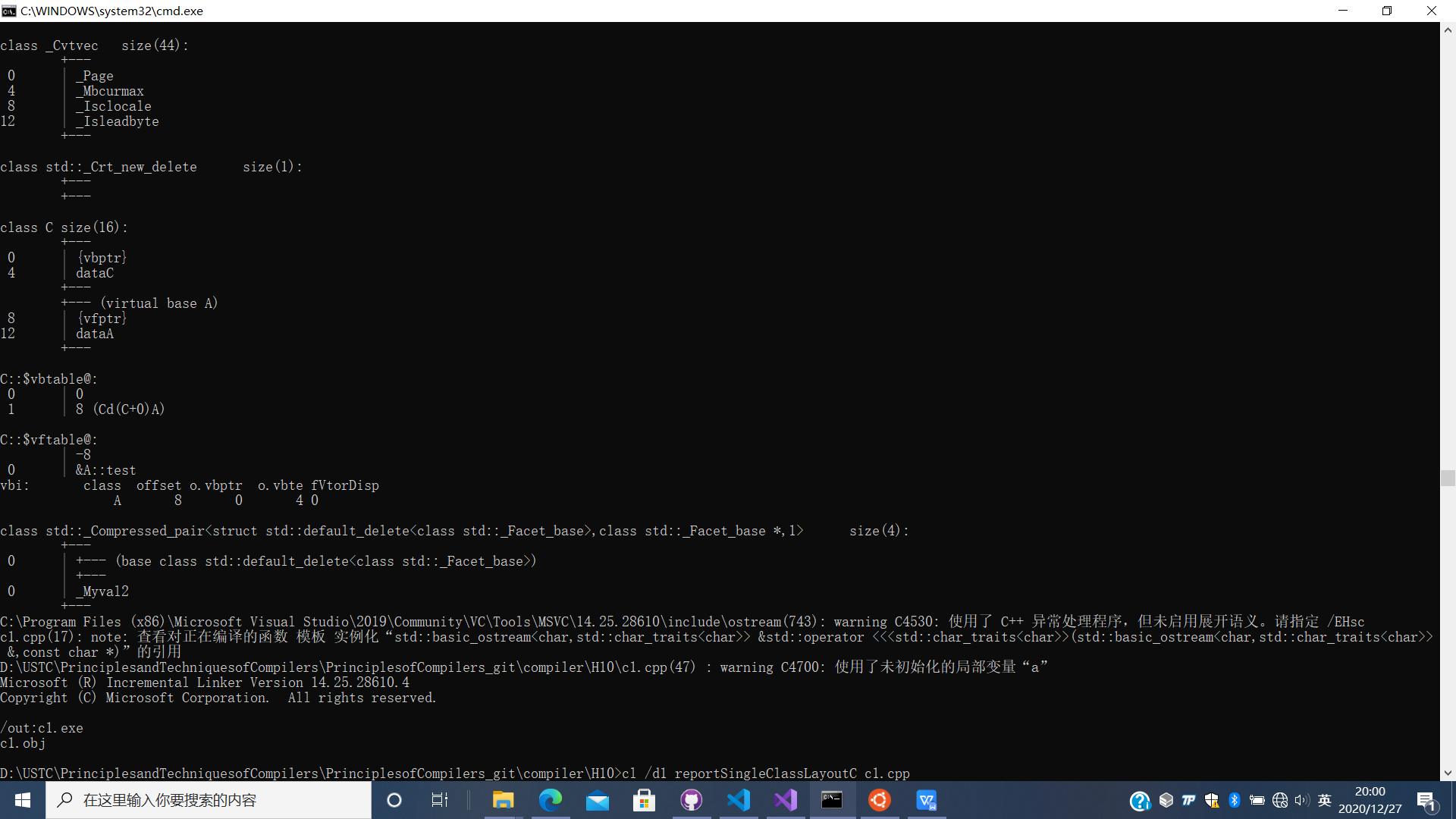
虚继承之所以能够实现在菱形继承中只保存一份共有基类，关键在于vbptr指针。Vbptr即(Virtual base table pointer)，该指针指向了一个虚表，虚表中第一项记录了vbptr与本类的偏移地址；第二项记录了vbptr到共有基类元素的偏移量(具体可见上图中的D::$vbtable$@B@和D::$vbtable$@C@)。

虚函数表指针vfptr与vbptr同理。由上图可知，D中保存的A部分有个虚函数表指针。表中指向了该类中的虚函数test，根据上图信息知，D中的test函数同B中的test函数。

B的对象布局：

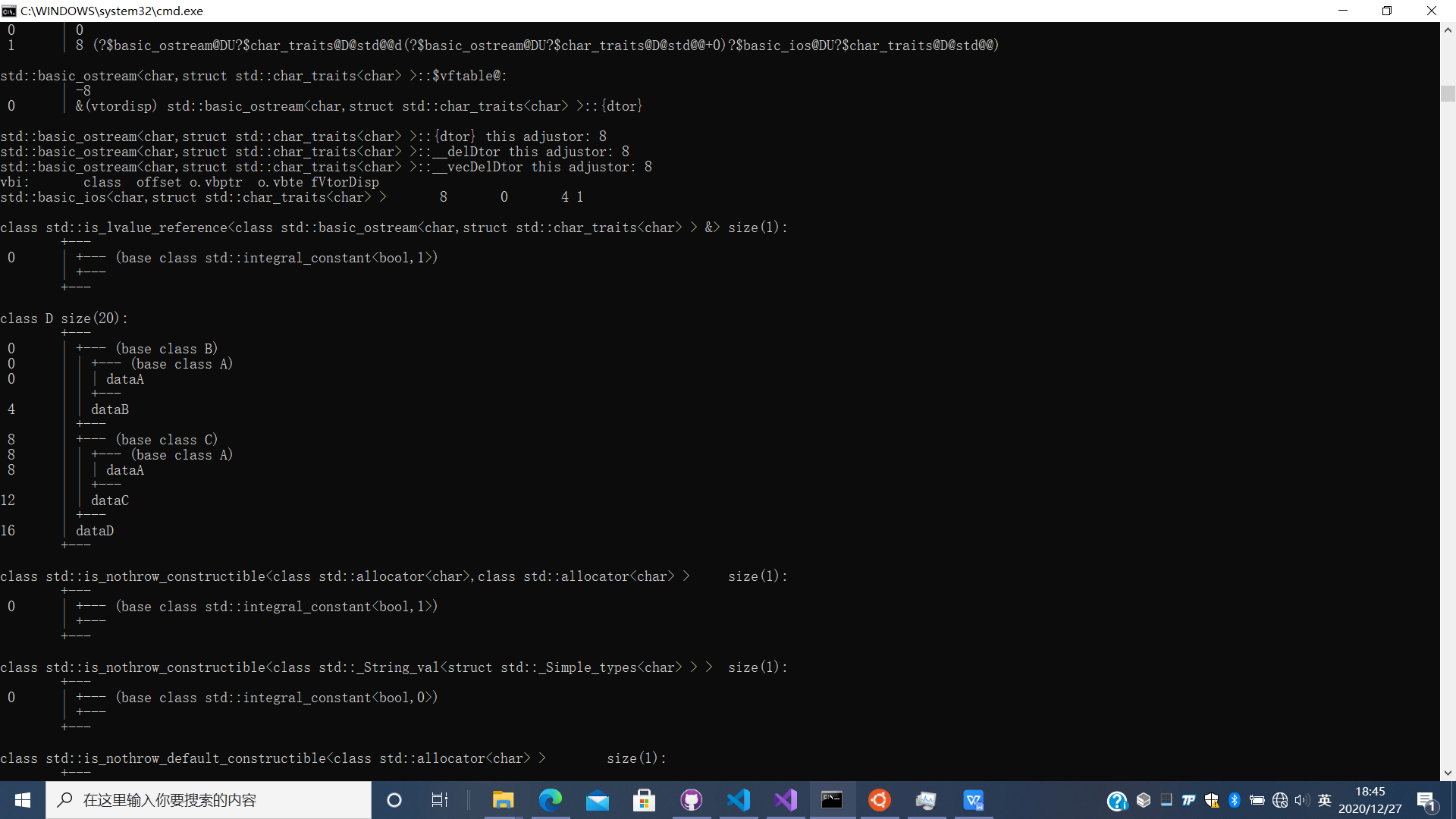


C的对象布局：

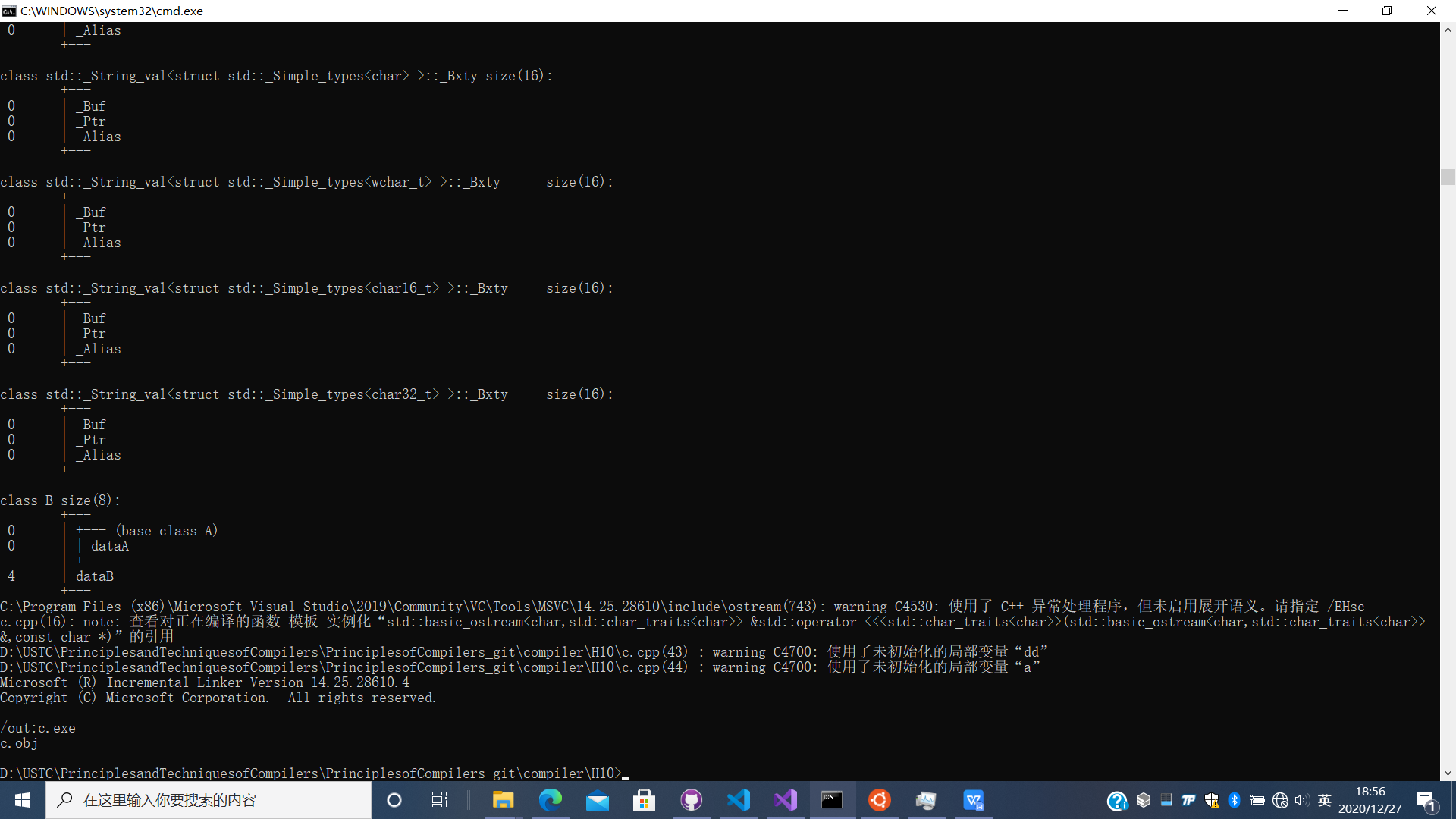


非虚继承：

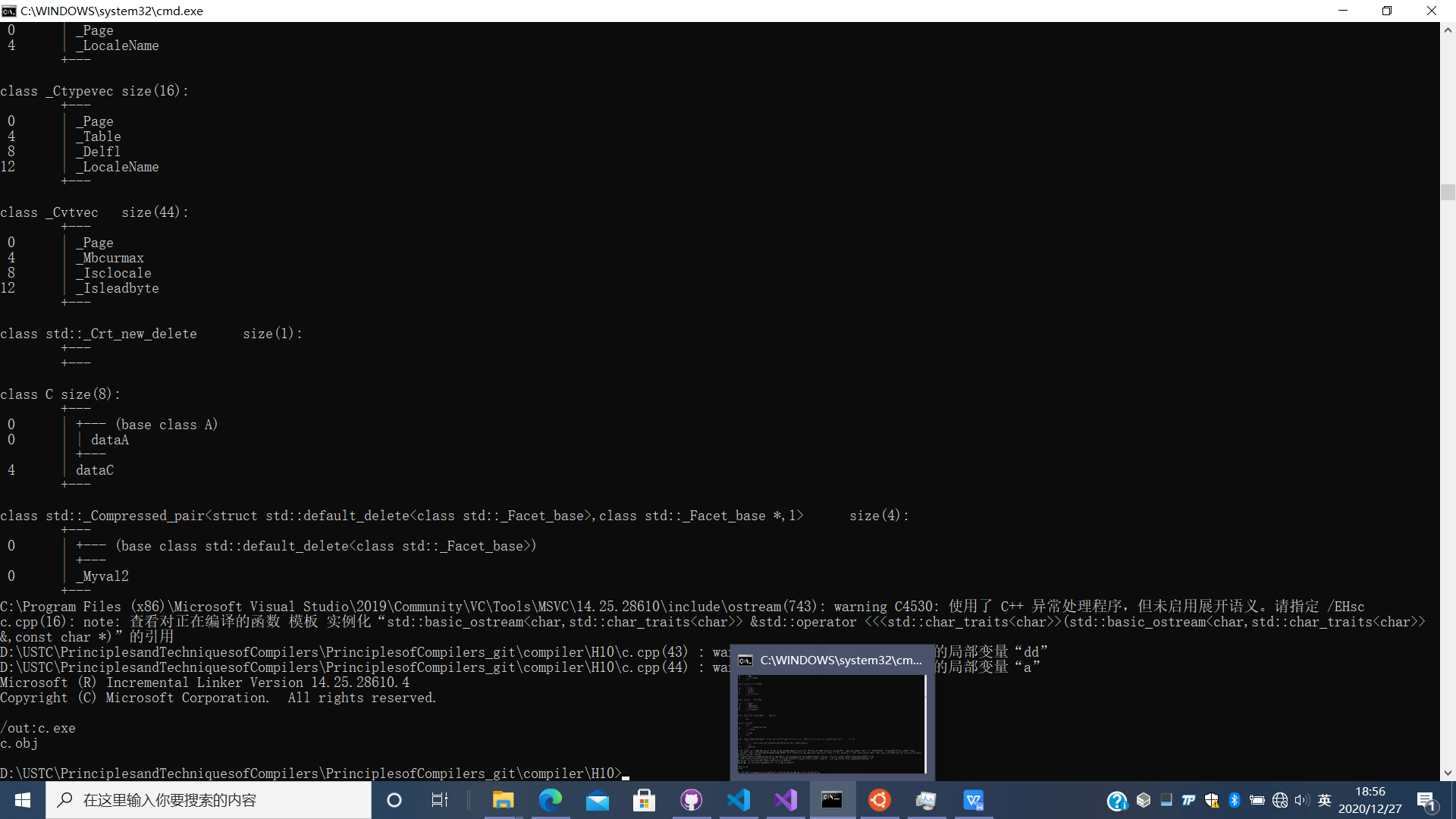
D的对象布局：



B的对象布局：



C的对象布局：



2.

1. 试从产生的汇编代码总结gcc处理inline的特征，内联是在编译的什么阶段被处理？

在未选择优化时，gcc按照对普通函数的处理方式调用f，而没有展开。在-O2优化中对内联函数进行了展开。内联是在编译的链接阶段被处理。

2）试说明编译器进行了哪些优化而得到带-O2选项生成的汇编码。

对f()进行了展开。展开后删除了f()中的死代码。运用了复写传播，减少了复写语句。公共子表达式a也用1进行了替换。

3）如果将inline.h第1行的static去掉，执行gcc inline.c inline1.c，产生如下错误，试说明原因，并指出这是在编译的哪个阶段产生的错误。

这是在链接的阶段产生的错误。

原因是在inline.c和inline1.c都include了inline.h，但没有使用static对函数f()的作用域加以限制，展开后，两个f()函数的作用域发生了重叠。故在链接inline.o和inline1.o时，报错出现f的重复定义。