1. 简单类的存贮空间

(1) 只有实例数据成员才占空间

(2) 静态数据成员和函数成员不占空间

```
class A {
        static long a;
        int  i;
public:
        int  k;
        static int b;
        friend int h() { return  -1; }
        int f() { return 0; }
        static int g() { return 1; }
        A() { }
} x;
```

x的存贮空间

int i	
int k	

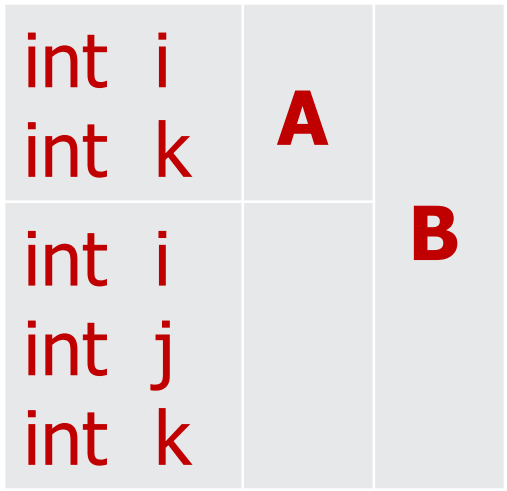
Question: 能否将h()写成{ return k; }?

2. 无虚函数的派生类存贮空间

派生类对象需要把基类空间包含进去

```
class A {
        static long a;
        int i;
public:
        int k;
        static int b;
        int f() { return 0; }
        static int g() { return 1; }
};
class B : A {
     int  i, j, k;
     static  int m;
} x;
```

x的存贮空间



如何访问x中A的数据成员？

3. 访问对象(无虚函数)中的数据成员


```
class A {
   static long a;
   int  i;
public:
   float  k;
   static int b;
   int f() { return 0; }
   static int g()
   { return 1; }
   A(int x,int y)
   { i = x; k = y + 0.5f; }
};
```

```
class B : A {
   int  i, j;
public:
   int  k;
   static int m;
   B(int x, int y, int z): A(x-1, y-1)
   { i = x; j = y; k = z; }
};
void main()
{
   B  b(1, 2, 3);
   int    *p1 = (int *)&b;
   float *p2 = (float *)(p1+1);
   int    *p3 = (int *)(p2+1);
   cout << p1[0] << p2[0] <<
   p3[0] << p3[1] << p3[2];
}
```

Results: 0 1.5 1 2 3

4. 有虚函数的简单类存贮空间

偏移 00 - 03：虚函数表VFT的地址
偏移04开始：非static的数据成员

```
class A {
    int  i, j;
    static  int  x, y;
    virtual void f( ) {  cout << "f()"; }
public:
    int  k;
    virtual void g() { cout << "g()"; }
    int h() { return i + k; }
};
```

//假设 sizeof(int) = 2

类A的存贮空间

偏移	内容
00-03h	VFT地址
03-04h	int i
05-06h	int j
07-08h	int k

VFT

偏移	内容
00-03h	f()的入口地址
04-07h	g()的入口地址

5. 有虚函数的派生类存贮空间 (1)

(1) 基类没有虚函数(派生类有)

```
class A {
    int  i;
    static  int  x, y;
public:
    int  k;
    int  h() { return i + k; }
};
class B: public A {
    int  i;
public:
    int  k;
    int  h() { return i + k; }
    virtual void f()  { cout << "B::f()"; }
    virtual void g() { cout << "B::g()"; }
};
```

//假设 sizeof(int) = 2

类B的存贮空间

偏移	内容
00-01h	int A::i
02-03h	int A::k
04-07h	B的VFT地址
08-09h	int B::i
0A-0Bh	int B::k

B的VFT

偏移	内容
00-03h	f()的入口地址
04-07h	g()的入口地址

5. 有虚函数的派生类存贮空间 (2)

(2) 基类有虚函数

```
class A {
    int  i;
    static  int  x, y;
public:
    int  k;
    virtual void f()  { cout << "A::f()"; }
    virtual void g() { cout << "A::g()"; }
};
class B: public A {
    int  i;
public:
    int  k;
    int  h()  { return i + k; }
    virtual void f()  { cout << "B::f()"; }
    virtual void k() { cout << "B::k()"; }
};
```

//假设 sizeof(int) = 2

类B的存贮空间

偏移	内容
00-04h	B的VFT地址
00-01h	int A::i
02-03h	int A::k
08-09h	int B::i
0A-0Bh	int B::k

B的VFT

偏移	内容
00-03h	B::f()的入口地址
04-07h	A::g()的入口地址
08-0Bh	B::k()的入口地址

5. 有虚函数的派生类存贮空间 (3)

派生类VFT构造方法：

(1) 将基类A的VFT复制到派生类B的VFT的开始处；

(2) 若派生类B对基类A的某个虚函数f()进行了重定义，则在刚拷贝的基类VFT中，用B::f()的地址替换A::f()函数的入口地址；

(3) 若在派生类B中定义了新的虚函数，则依次将这些新的虚函数的入口地址尾加到B的VFT。

6. 利用存贮空间访问变量和虚函数


```
class A {
    int  i;
public:
    virtual void f()
    { cout << "A::f \n"; }
    virtual void g()
    { cout << "A::g \n"; }
    A(int x) { i = x; }
};
class B : public A {
    int  i, j;
public:
    virtual void h()
    { cout << "B::h \n"; }
    void g()
    { cout << "B::g \n"; }
    B(int x,int y): A(x-1)
    { i = x; j = y; }
};
```

```
void main()
{
    B  b(1,2);
    int *v = (int *) ( (char *)&b +
                sizeof(void *) );
    printf("A::i=%d, B::i=%d,
    B::j=%d \n", v[0],v[1],v[2]);
    void **vfb = *(void ***)&b;
    for(int i = 0; i < 3; i++)
    {
        void (*f)() = (void (*)())vfb[i];
        f();
    }
}
```

A::i=0, B::i=1, B::j=2

A::f B::g B::h

能否直接取虚函数的地址，如 &b.f ？