1. sizeof a class

a. non-static class member variables

b. aligment

c. other overhead

1) virtual function: vptr

2) virtual base class: vbtr

Static member is not in the object but in global memory and all the objects share it.

If a class just an empty one that don’t have any item, the size of the class is 1. Because the compiler inserts a char to the class to make two different object have the unique address in the memory.

2. typedef for a class

Typedef declare should put on the beginning of the class that follows the Defensive programming rule.

typedef float len;

class Test

{

typedef int len;

private:

len item; /\*the len is resolved to int \*/

}

3. layout of the data of class

In the same access section each data member should consecutively in the memory that later ones in the high memory address. There is no overhead if there are several access section. But there is no rule for data member in different access section.

4. data member access

Static, non-static, single inherit and multiple inherit member data have the same speed with object and pointer access. But if a data memory inherit from a virtual base class in derived class the speed of object access is faster than pointer one.

5. single inheritance (non-virtual inheritance)

Class Derived : public Base{}

There is only one vptr in single inheritance, but the vptr table may be different. Vptr locate in the end of the first upper base class that have virtual function

Base Derv

+-----+ +-----+

|base | |base |

+-----+ +-----+

|vptr | |vptr |

+-----+ +-----+

|derv |

+-----+

6. multiple inherit (non-virtual inheritance)data pointer access

Class Derived: public Base1, public Base2{}

Derived d;

Base1 \*b1 = &d; /\*no modification in compiler\*/

Base2 \*b2 = &d; /\*b2 =(Base2\*)((char\*)Base1 + sizeof(Base2))\*/

Derived pointer point to the first Base class, there is no modification in compiler. But the following should add the offset that is the sum of front bases class size. But there is no overhead for the access speed, because all things are resolved in the compiler.

7. pointer to Data Member: type className::name;

Class T { Public: Int a; Int b; Int c; }/\*should be public\*/

Int T::\*p =&T::a; /\*value:0\*/

Int T::\*p =&T::b; /\*value:4\*/

Int T::\*p =&T::a; /\*value:c\*/

Value is the offset of the data member in the class. However, T t; &t.a presents the real address of the a member of the objects t.

Int T::\*p =&T::a;

T t; t.(\*p) means t.a;

T \*tn = &t; tn->(\*p) means tn->a;