#include <iostream>

using namespace std;

class CFather

{

private:

int father\_num;

public:

void fun(int a)

{

cout << "fun\_father\n";

}

CFather()

{

father\_num = 100;

cout << "CFather\n";

}

~CFather()

{

cout << "~CFather\n";

}

friend void show();

};

void show()

{

CFather fa;

cout << "Father\_Friend\n";

cout << fa.father\_num << endl;

}

class CSon : public CFather

{

public:

void fun()

{

cout << "fun\_son\n";

}

CSon()

{

cout << "CSon\n";

}

~CSon()

{

cout << "~CSon\n";

}

};

int main()

{

CSon person;

person.CFather::fun(1); 

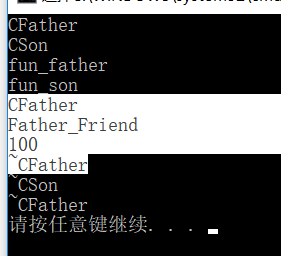
person.fun();

show();

//system("pause");

return 0;

}



编译结果：中间反白的是调用友元函数的结果，里面定义了一个CFather类，就会调用CFather类的构造函数和析构函数。