```
//class inheritance
#include <iostream>
using namespace std;

class A {//class A final {   …   With "final", class A cannot be inherited.
private://access specifier – the default access specifier is private.  This line is optional
//a private member can only be accessed by members of this class.
//or by friend functions
            int a;//a is a private member
protected:
//a protected member is a private member that allows inheritance
            int b;//b is a protected member
public:
//public members can be accessed by all functions in the programs
            int c;//c is a public member
            A(int i, int j) { b = i; c = j; }//consturctor

};
//A: base class
//B: derived class --- B is derived from A
//public level inheritance;
//Base class' private members are not inherited
//Base class' protected members are inherited as protected members in derived class
//Base class' public members are inherited as public members in derived class
//protected members remain protected in the derived class
//public members remain public in the derived class

class B : public A { //B has 5 data members:a1 (private), b, b1 (protected), c, c1(public)
//key word public, protected, or private in class inheritance is referred to as inheritance level
private: //this statement is optimal, because the default is private
            int a1; //a of class A is private and cannot be inherited
protected:
            int b1;//one more  b
//a protected member can only be accessed by members of this class
//and friend functions  -- the same as a private member
public:
            int c1;//one more c
            B(int p, int q, int i, int j) : A(i, j) { b1 = p; c1 = q; }//invoking constructor of base class

};

//C inherits from A using protected level
//private members of base class A cannot be inherited
//protected members of base class A is inherited by the derived class C as protected members
//public members of base class A is inherited by the derived class C as protected members
//C has one private member a2; has 3 protected members: b, c and b2; has 1 public member c2
//both public and protected members become protected in the derived class C
class C : protected A {
```

```
private:
        int a2;
protected:
        int b2;
public:
        int c2;

};




//D inherits from A using private level
//D has 3 private members a3, b, c
//D has 1 protected member b3
//D has 1 public member c3
//Both protected and public members of the base class A become private in the derived class D

class D : private A {//inheritance level:  Private; Private is default
private:
        int a3; //also b and c from class A
protected:
        int b3;
public:
        int c3;

};

class K: A {//default inheritance level is private
}



//class E: public A, protected AA
//multiple inheritance is supported in C++, but not Java


class F {//You can switch back and forth between different member modes.
        int k; //k is a private member
public:
        int k1;
protected:
        int k2;
private:
        int k3;
public:
        int k4;
};
```

```
class AA {//abstract class – not a completed class to be use

            int aa1;//private
protected:
            int aa2;
public:
            int aa3;
            int f1(int i) {
                        return i * aa1;
            }
            virtual int f2(int i) { //virtual member function can be redefined in the derived class
                        return i * aa1;
            }
            virtual int f3(int i) = 0; //pure virtual member function
                                                                                    //a class which contains pure virtual
member function(s) is called an abstract class
                                                                                    //and cannot be used
                                                                                    //Any derived class has to define the
function body for such a pure virtual member functions
};

class BB : public AA {

            int bb1;
protected:
            int bb2;
public:
            void set_bb1(int k) { bb1 = k; }
            int bb3;
            int f2(int i) {//virtual int f2(int i) is the same.  If "int f2 fianl {...", the f2 can no longer be
inherited.
                        return i + bb1;
                        //f2 is a virtual member function in the base class AA
                        //It can be redefined as needed
                        //It can also be directly used without being redefined.
            }
            virtual int f3(int i) {//Compiler requires you to define f3 because it is a pure virtual
member function in the base class AA
                                                                        //in the base class AA
                        return i * i * 2;
            }
};

//Everything we've discussed so far applies to struct
//Exception: default inheritance level: public
//default access specifier: public
```

```
int main() {

	AA* p1;
	BB o1;
	o1.set_bb1(10);
	p1 = &o1;
	p1->aa3 = 35;
	cout << p1->f2(5) << endl; //Note that, as an exception, p1 can access the virtual
funciton re-defined in BB.

	//p1->bb3 will give you error because p1 is defined for class AA
	//and it only allows access to those inherited from AA
	//Be more careful in this aspect in programming.

	//AA a; Error! beacuse AA is an abstract class and cannot be directly used yet
	BB b; //BB is not an abstract class
	getchar();
	getchar();
	return 0;
}
```