Έννοια	Τι είναι	Πού χρησιμοποιείται	Κύριο νόημα	Παράδειγμα
virtual	Λέξη-κλειδί για συναρτήσεις	Σε base classes	Επιτρέπει override & runtime binding	virtual void f();
Πολυμορφισμός	Ίδιο interface, διαφορετική συμπεριφορά	Με pointers/references	Διαφορετική υλοποίηση στην runtime	Shape* s = new Circle();
is-a	Σχέση κληρονομικότητας	Class inheritance	"είναι ένα"	Dog : public Animal
has-a	Σχέση σύνθεσης	Μέσα σε class	"έχει ένα"	Car has Engine
public	Access specifier	Μέλη class	Προσβάσιμα από παντού	public: int x;
private	Access specifier	Μέλη class	Μόνο μέσα στην class	private: int x;
protected	Access specifier	Μέλη class	Class + derived classes	protected: int x;
final	Λέξη-κλειδί	Σε class ή method	Απαγορεύει override/κληρονομικότητα	void f() final;
friend	Ειδική πρόσβαση	Functions/classes	Σπάει το encapsulation	friend void func();

- **virtual + πολυμορφισμός** → δουλεύουν μαζί (runtime behavior)
- **is-a vs has-a** →
  - is-a = inheritance
  - has-a = composition
- **public/private/protected** → έλεγχος πρόσβασης
- **final** → "σταματά εδώ"
- **friend** → δίνει "VIP access" σε άλλη function/class

1. Κληρονομικότητα (Inheritance)

```
class Base {};
class Derived : public Base {};
```

✓ is-a σχέση

Dog is-a Animal

2. Σύνθεση (Composition)

```
class Engine {};

class Car {
   Engine e;
};
```

has-a σχέση

Car has-a Engine

3. Access Specifiers

Modifier	Πρόσβαση
public	παντού
private	μόνο μέσα στην class
protected	class + derived

4. Virtual & Polymorphism

```
class Base {
public:
   virtual void f() { cout << "Base"; }
};

class Derived : public Base {
public:
   void f() override { cout << "Derived"; }
};
Base* b = new Derived();
b->f(); // Derived
```

✓ Runtime polymorphism
✓ Dynamic binding

5. Abstract Class

```
class Shape {
public:
   virtual void draw() = 0;
};
```

✓ Δεν δημιουργείται object
✓ Μόνο inheritance

6. Multiple Inheritance

```
class A {};
class B {};

class C : public A, public B {};
```

7. Diamond Problem

```
class A {};
class B : virtual public A {};
class C : virtual public A {};
class D : public B, public C {};
```

✓ Αποφυγή διπλού A

8. Virtual Destructor

```
class Base {
public:
   virtual ~Base() {}
};
```

✓ Απαραίτητο για polymorphism

9. final

```
class A final {};
void f() final;
```

✓ Δεν γίνεται override / inheritance

10. friend

```
class A {
   friend void func();
};
```

✓ Πρόσβαση σε private μέλη

Διαγράμματα

is-a (Inheritance)

Animal
↑
Dog

has-a (Composition)

Car ——► Engine

Πολυμορφισμός

Base*
▼
Derived object
Καλείται η Derived συνάρτηση

Multiple Inheritance

```
 A   B
  \ /
   \/
    C
```

Diamond Problem

```
    A
   / \
  B   C
   \ /
    D
```

✓ λύση → virtual

Access Levels

public → όλοι
protected → class + derived
private → μόνο class

Γρήγορο Mnemonic

- **is-a** → inheritance
- **has-a** → composition
- **virtual** → polymorphism
- **protected** → "κληρονόμοι μόνο"

- **final** → STOP
- **friend** → bypass encapsulation

Άσκηση 1 – Απλή κληρονομικότητα (is-a)
Εκφώνηση: Δημιούργησε μια κλάση Animal με μέθοδο speak(). Δημιούργησε κλάση Dog που κληρονομεί από Animal και κάνει override τη speak().
Hint: Χρησιμοποίησε virtual στη βάση και override στην παράγωγη.

```
#include <iostream>
using namespace std;

class Animal {
public:
    virtual void speak() {
        cout << "Animal makes a sound\n";
    }
};

class Dog : public Animal {
public:
    void speak() override {
        cout << "Dog barks\n";
    }
};

int main() {
    Dog d;
    d.speak();
}
```

Επεξήγηση:
- **Dog is-a Animal**
- **Το virtual επιτρέπει πολυμορφισμό.**

Άσκηση 2 – is-a σχέση με constructor
Εκφώνηση: Κλάση Person με όνομα. Κλάση Student που προσθέτει βαθμό.
Hint:Κάλεσε constructor της βάσης με :.

```
class Person {
protected:
    string name;
public:
    Person(string n) : name(n) {}
};
```

```
class Student : public Person {
    int grade;
public:
    Student(string n, int g) : Person(n), grade(g) {}
};
```

Επεξήγηση: Ο constructor της βάσης καλείται πρώτος.

Άσκηση 3 – has-a σχέση
Εκφώνηση: Κλάση Engine και κλάση Car που έχει engine.
Hint: Composition → αντικείμενο μέσα σε άλλο.

```
class Engine {
public:
    void start() {
        cout << "Engine starts\n";
    }
};

class Car {
    Engine engine;
public:
    void startCar() {
        engine.start();
        cout << "Car is ready\n";
    }
};
```

Επεξήγηση:

- **Car has-a Engine**
- **Δεν είναι inheritance.**

Άσκηση 4 – override & polymorphism
Εκφώνηση: Δημιούργησε Shape με area(), και Circle.
Hint: Χρησιμοποίησε pointer σε βάση.

```
class Shape {
public:
    virtual double area() { return 0; }
};

class Circle : public Shape {
    double r;
public:
    Circle(double r) : r(r) {}
```

```
    double area() override {
        return 3.14 * r * r;
    }
};
```

Επεξήγηση: Runtime polymorphism.

Άσκηση 5 – πολλαπλή κληρονομικότητα
Εκφώνηση: Κλάση Printer και Scanner. Δημιούργησε AllInOne.
Hint: Χρησιμοποίησε : με πολλές βάσεις.

```
class Printer {
public:
    void print() { cout << "Printing\n"; }
};

class Scanner {
public:
    void scan() { cout << "Scanning\n"; }
};

class AllInOne : public Printer, public Scanner {};
```

Επεξήγηση: Πολλαπλή κληρονομικότητα → συνδυασμός δυνατοτήτων.

Άσκηση 6 – virtual destructor
Εκφώνηση: Γιατί χρειάζεται virtual destructor;
Hint: Χρήση pointer σε base.

```
class Base {
public:
    virtual ~Base() {
        cout << "Base destroyed\n";
    }
};

class Derived : public Base {
public:
```

```
    ~Derived() {
        cout << "Derived destroyed\n";
    }
};
```

Επεξήγηση: Αποφεύγει memory leaks.

Άσκηση 7 – protected μέλη
Εκφώνηση: Δείξε πώς ένα protected μέλος χρησιμοποιείται σε derived.

```
class Base {
protected:
    int x;
};

class Derived : public Base {
public:
    void setX(int v) { x = v; }
};
```

Επεξήγηση: protected → προσβάσιμο από derived.

Άσκηση 8 – abstract class
Εκφώνηση: Δημιούργησε abstract class Vehicle.
Hint: Pure virtual function.

```
class Vehicle {
public:
    virtual void move() = 0;
};

class Car : public Vehicle {
public:
    void move() override {
        cout << "Car moves\n";
    }
};
```

Επεξήγηση: Δεν μπορείς να κάνεις instantiate abstract class.

Άσκηση 9 – diamond problem
Εκφώνηση: Δείξε το diamond problem και λύσε το.
Hint: Χρησιμοποίησε virtual.

```
class A {
public:
    int x;
};

class B : virtual public A {};
class C : virtual public A {};

class D : public B, public C {};
```

Επεξήγηση: Αποφεύγεται διπλό αντίγραφο της A.

Άσκηση 10 – συνδυασμός has-a και is-a
Εκφώνηση: Κλάση Employee και Manager (is-a), αλλά ο Manager έχει Team.

```
class Team {
public:
    int members;
};

class Employee {
public:
    string name;
};

class Manager : public Employee {
    Team team;
};
```

Επεξήγηση:

- **Manager is-a Employee**
- **Manager has-a Team**